Model predictive control of offshore wind turbine

Author
ADRIAN B. SKIBELID

Supervisors
LEIF E. ANDERSSON
LARS S. IMSLAND
KONSTANZE KÖLLE

December 15, 2019

NTNU – Trondheim
Norwegian University of
Science and Technology

Abstract

The work presented in this thesis develops a Model Predictive Control (MPC) controller for SINTEF's offshore wind turbine simulator, STAS. STAS is a frequency domain simulator that is not optimized for time simulation and gives out linearized systems for two linearization parameters; mean wind speed and percent of rated power that the turbine is operating at. The MPC is first developed for only one such linear system. The earliest MPC iterations controlled the generator speed and the collective blade pitch in a step-wise manner, but this caused harmonics to be added to the system. Attemps at reducing the harmonics were made by letting the MPC control a low-passed input, then linearly varying inputs and then low-passed, linearly varying inputs, before finally the MPC was made to control set points of a Proportional-Integral (PI) controller.

The main objective is to reduce the power tracking error, which the MPC is shown to do. An attempt is made at reducing the fatigue on the structure by controlling the change in the moments of parts of the turbine to zero. This did however lead to a rather poor power tracking and not much reduction in the number of zero crossings of the derivative, which is the usual reason for fatigue on moving structures.

To allow more steps to be taken in the prediction of the MPC whilst keeping the system real time capable, the system is reduced to fewer states. This is first done with a balanced reduction, achieving a reduction from 103 states to 62. Another attempt is made via Principal Component Analysis (PCA), achieving the same performance with only 24 states.

Better power tracking is sought after and achieved by simulation of LIght Detection And Ranging (LIDAR) measurements of the wind speed of air upstream of the turbine. To test the system performance two noise models are implemented; one is a simple additive Gaussian noise model and the other is based on a finite difference solution to the transport equation. With perfect measurements the feed forward MPC lead to a large reduction in Root Mean Square (RMS) error between desired and actual power, and a smaller, yet significant, improvement is achieved even in the presence of a decent amount of noise on the measurements.

To have the system behave more realistically away from the linearization point of the STAS model, a Linear Parameter-Varying (LPV) system is made by interpolation of linearized systems for three different linearization parameters. The system is shown to have worse power tracking than the PI controller when using one static projection matrix to reduce the system. Better performance than the PI controller is obtained when each linearized system is reduced separately and transformed into a form where they can be interpolated. Running the feed forward MPC on the full interpolated system with predictions using the interpolated reduced system gave lower RMS error than the PI controller, showing the promise of feed forward MPC for better power control and energy capture from offshore wind turbines.

Oppsummering

Arbeidet presentert i denne oppgaven utvikler en Model Predictive Control (MPC) kontroller for STAS, SINTEF sin simulator for offshore vindturbiner. STAS er en simulator i frekvensdomenet som ikke er optimert for tidssimulering. Den gir ut lineariserte systemer for to parametere; gjennomsnittlig vindhastighet og prosent av maks kraftproduksjon som vindmøllen opererer ved. En MPC er først utviklet for kun ett slikt linearisert system. De tidligste utgavene av MPC-en kontrollerte generatorhastigheten og den felles pitchen til bladene på en stegvis måte. Dette førte til høyfrekvent støy på enkelte tilstander, så det ble lett etter en bedre løsning, først igjennom å lavpassfiltrere inngangene, så ved å la MPC-en kontrollere lineært varierende innganger. Deretter ble det testet med lavpassfiltrerte, lineært varierende innganger, før det til slutt ble testet å la MPC-en kontrollere settpunktene til en Proportional-Integral (PI)-kontroller.

Hovedmålet var å redusere avvik i kraftproduksjonen fra en referanse, noe MPC-en viste at den klarte. Et forsøk på å redusere slitasje på vindturbinen ble gjort igjennom å styre endringen i enkelte strukturmomenter til null. Dette førte til en kraftig redusert kraftkontroll uten å vise til noen særlig reduksjon i antall nullkrysninger for den deriverte til momentene, noe som er blitt vist å være hovedkilden til slitasje på bevegelige strukturer.

For å tillate en finere diskretisering i prediksjonssteget til MPC-en uten å gå på bekostning av sanntidsegenskapene til kontrolleren ble det lineære systemet redusert til færre tilstander. Dette ble først gjort igjennom en balansert reduksjon, noe som ga en reduksjon fra 103 til 62 tilstander. Et nytt forsøkt via Principal Component Analysis (PCA) ga en reduksjon til 24 tilstander.

Bedre følging av kraftreferansen ble vist igjennom en foroverkoblet MPC basert på simulerte LIght Detection And Ranging (LIDAR)-målinger av vindhastigheten foran vindturbinen. For å teste ytelsen til systemet ble to forskjellige støymodeller implementert. Den ene var simpel hvit støy på hver måling, mens den andre brukte en finite difference løsning av transportlikningen. Med perfekte LIDAR-målinger ga systemet en stor reduksjon i Root Mean Square (RMS) feil mellom ønsket og faktisk produsert kraft. En mindre, men betydelig forbedring ble sett selv ved relativt mye støy på målingene.

For å få et mer realistisk system når tilstandene er langt unna lineariseringspunktet til STAS-modellen ble det lagd et Linear Parameter-Varying (LPV)-system via interpolering av lineariserte systemer for tre forskjellige parametere. Systemet viste å ha dårligere kraftfølging enn den PI-kontrolleren når det kun ble brukt én statisk projeksjonsmatrise for å redusere system. Bedre ytelse enn PI-kontrolleren ble derimot oppnådd når hvert lineariserte system ble redusert separat og transformert til en form hvor de kunne bli interpolert. Når MPC-en ble kjørt på det fulle interpolerte system med prediksjoner ved bruk av det interpolerte reduserte systemet ga det lavere RMS-feil enn PI-kontrolleren. Dette gir håp om at foroverkoblet MPC kan bli brukt til å redusere feil i kraftfølging og øke energifangsten til offshore vindturbiner.

Acknowledgements

Several people were involved and helped with this thesis.

I would first and foremost like to thank my super hero supervisor, Leif Erik Andersson, for his patient, ever-present support. He was there whenever I got stuck or needed someone to discuss with, and have gone above and beyond to help and assist me, always willing to provide smart, on-point input. Leif also provided the initial CasADI code, which kick-started the thesis and made it a lot easier to wrap my head around the framework.

I am very grateful to Lars Struen Imsland for having the time to help with the big picture of the research even in his hectic days as the temporary head of the Department of Engineering Cybernetics.

My sincere thanks go out to Konstanze Kölle for her patient explanation of the intricate inner workings of the STAS simulator, as well as general feedback.

I am also indebted to Morten Dinhoff Pedersen for providing me with the main idea behind the finite difference noise model for the LIDAR simulation, and Sebastien Gros for in-debt feedback about issues with lack of controllability of the turbine moments.

Finally I would like to thank Sanne Helvig for continuous support, proof reading and for always having the time to listen to me ramble about problems I encountered.

Table of Contents

1 Introduction **1**
- 1.1 Offshore wind turbines . . . 1
- 1.2 Previous work . . . 1
- 1.3 Problem formulation . . . 2
- 1.4 Contributions . . . 2

2 Background **4**
- 2.1 Wind turbine theory . . . 4
  - 2.1.1 Nomenclature . . . 4
  - 2.1.2 Aerodynamic subsystem . . . 6
  - 2.1.3 Mechanical subsystem and bending modes . . . 7
  - 2.1.4 Power generator unit . . . 7
  - 2.1.5 Pitch subsystem . . . 8
  - 2.1.6 Wind model . . . 8
  - 2.1.7 Upstream wind speed profile . . . 9
- 2.2 Model Predictive Control . . . 9
- 2.3 STAS . . . 13
- 2.4 Model reduction . . . 14
  - 2.4.1 Principal Component Analysis . . . 15
  - 2.4.2 Balanced Truncation . . . 16
- 2.5 Interpolation of Linear Time-Invariant systems . . . 17
  - 2.5.1 Coherence of LTI systems . . . 17
  - 2.5.2 Transforming the reduced systems to a common basis . . . 18
  - 2.5.3 Independent interpolation of elements . . . 22

3 Method **23**
- 3.1 Linear MPC . . . 23
- 3.2 Model reduction . . . 24
  - 3.2.1 Controlling PI set points . . . 25
- 3.3 Reducing moment derivatives . . . 25

3.4 LIDAR feed forward 25
3.4.1 Measurement noise and filtering 29
3.4.2 Finite difference noise model 29
3.5 Linear Parameter-Varying System 31

4 Results 33
4.1 Full linear system 33
4.2 System reduction 36
4.3 Feed forward wind measurements 38
4.4 Interpolation 44
4.4.1 Interpolation of full system 44
4.4.2 Interpolation of reduced system 45
4.5 Moment reduction 47

5 Discussion 49
5.1 The importance of model reduction 49
5.2 Controlling PI set points 49
5.3 LIDAR cost/profit analysis 49
5.4 Accuracy of simulated LIDAR measurements 50
5.5 Discussion of control of LPV system 51
5.6 STAS not directly suitable for fatigue reduction using MPC 51

6 Conclusion 52

7 Future work 54

Glossary 60

List of Acronyms 62

Figures

2.1 Illustration of wind turbine with basic nomenclature 5
2.2 Subsystem level block diagram of wind turbine 5
2.3 Relevant mode shapes of a wind turbine. 7
2.4 Illustration of how vortex sheet theory models the velocity decrease as wind approaches a turbine 10
3.1 Side view of a simplified version of the wind measurement field from a hub mounted LIDAR 26
3.2 Wind velocity as a function of distance from rotor and time, without any noise 31
4.1 System response of linear MPC using full system for predictions, with step inputs to the system 34
4.2 System response of linear MPC using full system for predictions, with the MPC controlling filtered inputs 35
4.3 System response of linear MPC using full system for predictions, with linearly varying inputs 35
4.4 System response of linear MPC using full system for predictions, with filtered, linearly changing inputs 36
4.5 System response of MPC with predictions using system reduced to 62 states with balanced reduction 37
4.6 System response of MPC with predictions using system reduced to 22 states with PCA 37
4.7 System response of MPC with predictions using system reduced to 24 states with PCA, controlling set points of PI controller 38
4.8 PI controller responding to ramp wind input. 39
4.9 MPC without feed forward responding to ramp wind input 40
4.10 Feed forward MPC with noiseless LIDAR measurements responding to ramp wind input 40
4.11 Feed forward MPC with LIDAR measurements with additive Gaussian noise, responding to ramp wind input 41

4.12 Pure PI controller responding to the same input wind as the feed forward MPC controllers seen in figures 4.14 and 4.16 42
4.13 Measured wind velocity as a function of distance from rotor and time, with Gaussian noise on inlet 42
4.14 Feed forward MPC with LIDAR measurements from the wind profile seen in 4.13 43
4.15 Measured wind velocity as a function of distance from rotor and time, with Gaussian noise on inlet, process noise between wind states and an added gust 43
4.16 Feed forward MPC with LIDAR measurements from the wind profile seen in 4.15 44
4.17 System response when running MPC using the interpolated full system with only one static projection matrix 46
4.18 System response when using a PI controller on the interpolated full system 46
4.19 System response when running MPC using interpolated reduced system . 47
4.20 System response to a step input in wind without cost on any of the moment derivatives 48
4.21 System response to a step input in wind with cost on the moment derivatives of the blade root and tower 48

List of Tables

4.1 Root mean square error between actual and desired power for different controllers and noise models, showcasing the benefits of LIDAR feed forward. 38
4.2 Root mean square error between actual and desired power, comparing pure PI control over feed forward MPC with finite difference noise. 41
4.3 Root mean square error between actual and desired power for three different controllers, controlling the interpolated full system. 45
4.4 Root mean square error between actual and desired states with and without a cost on the blade root and tower moment derivatives. 47

Chapter 1

Introduction

This thesis was written in collaboration with SINTEF, using their wind turbine simulator, STAS. It aims to use Model Predictive Control (MPC) and feed-forward wind measurements to improve power tracking and possibly reduce structural fatigue.

1.1 Offshore wind turbines

Offshore wind is growing like never before [10]. Many large offshore projects are currently in operation, like the Walney Extension and the London Array, showing great promise. Offshore wind already has a strong foothold in Europe with more than 18 GW installed capacity [24], and global potential to reach more than 100 GW by 2030. Hywind, Equinor's floating wind turbine project outside Scotland, consists of five 6 MW turbines with a total installed capacity of 30 MW and powers around 22,000 households [9]. It indicates that it will be possible to develop wind farms in deeper waters, opening up a lot more area to be used for energy production.

This expansion of the wind sector makes it crucial that more advanced control systems for wind turbines are developed, in order to optimize energy capture whilst increasing the life time of the turbines. With higher penetration of renewable energy, better power tracking of wind turbines will also become increasingly important to maintain stability of grid frequency.

1.2 Previous work

There have been some studies on MPC for wind turbines. In [12] a MPC for a 10 state nonlinear model of a wind turbine is developed. It linearizes the model at each step, and focuses on constraint handling in all operating regions of the wind turbine, rather than improved power tracking abilities.

In [27], benefits are attained with perfect wind field preview 5 seconds ahead of time on simulated data using FAST [28], but the controller is not tested with more realistic LIght Detection And Ranging (LIDAR) specs or noisy measurements. It also found that the benefit of preview MPC was mainly seen in the full-load operating region.

More realistic LIDAR measurements where used in [16] and [23], and concluded that, even in the presence of imperfect wind speed measurements, large reductions in both extreme and fatigue loads were possible. The latter of these reported promising load reduction up to 50 % for extreme gusts and 30 % for lifetime fatigue loads without negative impact on overall energy production.

[15] develops a feed forward MPC using perfect preview of the future rotor plane average free stream wind speed. The authors focus, as with this thesis, on the full-load operating region, because the potential gains from improved performance in partial load operation are small, as shown in [23]. The cost function includes costs on state deviations, costs on inputs and costs on the final states deviation from a desired set point, but not on change in input. Even though it uses perfect wind preview to get the results, it is argued that as long as the upwind information provides any additional information above just assuming the wind to remain constant, the feed forward component of the controller can be expected to improve control performance. It furthermore proves good performance and constraint handling, even in the presence of modeling errors and stochastic turbulence. A large step towards robust MPC is taken by showing that it can handle many different scenarios and operating conditions. The controller can make extreme scenarios like shut downs more rare, but also reduce loads and fatigue if they happen. The paper concludes that MPC gives hope for a unified way of achieving the goals of a wind turbine controller while obeying constraints, in all operating regions and in the presence of modeling errors.

1.3 Problem formulation

Develop a MPC controller for the STAS model, and investigate if it can improve power tracking and reduce fatigue on the turbine. Develop a Linear Parameter-Varying (LPV) system from interpolation of STAS linearizations and test performance of the controller on this system.

1.4 Contributions

A linear MPC that showed improved power tracking of the STAS model was developed and tested. A linearized version of STAS was reduced from 103 to 24 states to speed up computation time of the MPC and give room for more advanced control schemes. LIDAR feed forward MPC was used to improve power tracking, with noisy measurements made to be similar to the specifications of real LIDARs. The possibility of reducing fatigue on the turbine structure with MPC was investigated, but concluded that it was not promising, possibly due to the structure of the STAS model or due to the much faster dynamics of the turbine moments compared to the MPC step size. A LPV system was developed from interpolation of linearizations of the STAS model and from a set of reduced linearized systems. A MPC was made to run on the interpolated reduced system and shown to give better power tracking than a simple Proportional-Integral (PI) controller when applied to simulations of the full interpolated system.

The work done in this thesis will be presented in January at the EERA DeepWind'2020 conference [26] in Trondheim.

Chapter 2

Background

In this section the needed theory and nomenclature is explained. It first covers wind turbines in general, with standard nomenclature and modeling, as well as how one models the wind close to the turbine. Next the theory behind MPC is covered, before a brief introduction to STAS is given. After this, model reduction is explained, and then finally how to create a LPV system by interpolation of several Linear Time-Invariant (LTI) systems.

2.1 Wind turbine theory

In the following section the wind turbine and a standard way of modeling it is described. It starts with simple nomenclature, before delving into the different subsystems of the turbine and finally explaining some common wind phenomena. Most of the definitions and models are from [3].

2.1.1 Nomenclature

The wind turbine is composed of several different parts, illustrated in figure 2.1. The nacelle is the part containing the generator, flywheel and gearbox. This is connected through the drive shaft to the rotor, which is composed of the hub and blade, as well as pitch actuators inside the hub that control the pitch of the blades. The pitch subsystem is explained further in section 2.1.5. The nacelle sits on top of the tower, which is connected to the foundation. In the interface between tower and nacelle sits a yaw motor, that makes it possible to turn the wind turbine to face the wind direction. In this thesis the yaw controller is ignored and the turbine is assumed to face the wind at all times.

A wind turbine can be viewed as a series of interconnected subsystems. This is usually done by dividing the system into three parts; the aerodynamic subsystem, the mechanical subsystem and the power generator unit. A block diagram of this way of looking at the turbine is illustrated in figure 2.2.

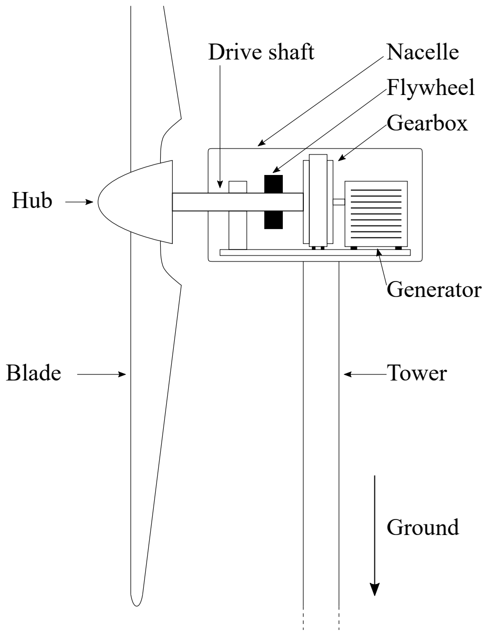

Figure 2.1: Illustration of wind turbine with basic nomenclature.

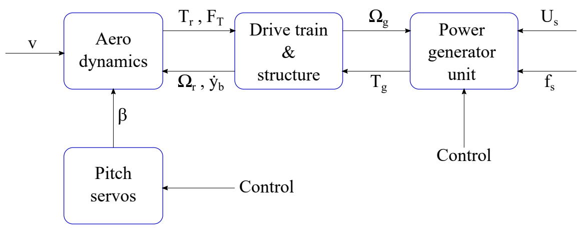

Figure 2.2: Subsystem level block diagram of wind turbine.

The aerodynamic subsystem models how the rotor generates a torque, T_r, and a force, F_T, on the drive shaft. This torque and force is dependent on the current wind velocity field at the rotor plane, $\vec{V}$, the pitch of the blades, β, and the rotational velocity of the rotor, Ω_r. They also depend on the axial displacement of the blades, y_b, caused by flapping and tower bending, which will be explained later.

The mechanical subsystem includes the rotor, hub, drive shaft and the mechanical parts of the generator, as well as the tower and foundation. It is usually further divided into two parts; the drive train and the structure. The drive train transfers the aerodynamic torque on the blades to the generator shaft. The structure contains the tower and foundation and is excited by the thrust force. The mechanical subsystem determines the rotational speed of the generator, Ω_g, as a function of the aerodynamic torque and force as well as the torque coming from the power generator unit, T_g.

This leads to the final subsystem, the power generator unit. The power generator unit is the electrical part of the generator, which converts the rotational speed of the generator into electrical power.

The torque coming from the generator and the pitch of the blades are the controllable quantities in the turbine, as well as the yaw of the nacelle.

2.1.2 Aerodynamic subsystem

An approach to model the thrust force, F_T, and the aerodynamic torque, T_r, is to first lump the continuous three dimensional wind field into a discrete number of forces acting on the rotor blades. These forces then give rise to F_T and T_r through a stationary mapping

$$\begin{bmatrix} F_T \\ T_r \end{bmatrix} = \begin{bmatrix} \frac{\rho \pi R^2}{2} C_T(\frac{\Omega_r R}{V_e}, \beta) v_e^2 \\ \frac{\rho \pi R^3}{2} C_Q(\frac{\Omega_r R}{V_e}, \beta) v_e^2 \end{bmatrix}. \tag{2.1}$$

Here ρ is the air density, R is the length of the blades, v_e is the wind speed relative to the rotor,

$$v_e = v_m - \dot{y}_b, \tag{2.2}$$

i.e. the difference between actual mean wind speed, v_m, and the change in axial displacement of the blades, y_b, caused by flapping and tower bending. The aerodynamic torque and the thrust force can be seen to be expressions depending on the two functions $C_T(\lambda, \beta)$ and $C_Q(\lambda, \beta)$, respectively. Here $\lambda = \frac{\Omega_r R}{v_e}$ is called the Tip Speed Ratio (TSR). C_T is a nonlinear function representing the drag coefficient of the blades, while C_Q is a nonlinear function representing the lift coefficient. Both these functions are typically found empirically in a wind tunnel.

A typical feature of the drag and lift functions is that they usually have a single maxima that is close to, or at, zero pitch. The drag function has its maximum at a lower TSR than the lift function.

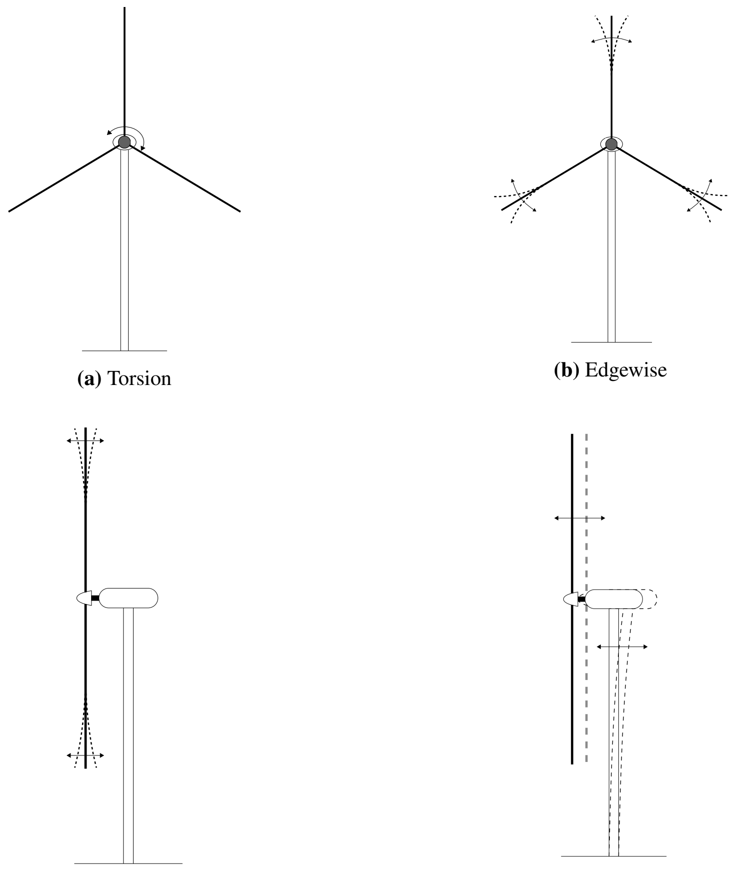
(a) Torsion **(b)** Edgewise **(c)** Flapwise **(d)** Tower bending

Figure 2.3: Relevant mode shapes of a wind turbine.

2.1.3 Mechanical subsystem and bending modes

A typical model for the mechanical subsystem is derived using the Multibody System (MBS) approach, and one such model can be found in [3]. It is a control-oriented approach to modeling that only focuses on the modes of the system that are relevant to control. The mechanical degrees of freedom that are most affecting the fatigue of the wind turbine, and thus most important for this thesis, are shown in figure 2.3. These are mostly self-explanatory, except maybe for the first one, torsion. Torsion refers to the twisting of the drive shaft because of a difference in the angle of the rotor and the angle of the generator.

2.1.4 Power generator unit

Since the induction generators used in modern wind turbines have much faster dynamics than the rest of the wind turbine systems, a steady-state-model of the power generator unit will be sufficient. Most modern wind turbines use a squirrel-cage induction generator, that is decoupled from the grid frequency and voltage via a DC-DC converter.

The frequency and voltage operation points of the generator can be controlled. The most common control scheme is to keep the flux constant while controlling the rotational speed of the generator. This is done by keeping the ratio between voltage, U_g and frequency, f_g on the generator side constant. The effect of this is that the rotational speed of the generator at no-load condition, Ω_z, can be seen as control input, which gives the following linear approximation of the torque characteristic

$$T_g = B_g(\Omega_g - \Omega_z). \tag{2.3}$$

No-load condition here means $T_g = 0$.

2.1.5 Pitch subsystem

Modern wind turbines have blades with adjustable pitch. These are sometimes controlled by a single motor controlling all blades to the same pitch, but most newer turbines have one motor for each blade, giving control of each blades pitch separately.

The motors are nonlinear servo motors, but operation in their linear region can be described by

$$\dot{\beta} = -\frac{1}{\tau}\beta + \frac{1}{\tau}\beta_d, \tag{2.4}$$

where β and β_d are the actual and desired pitch angles, respectively. According to [3] β varies from $-2°$ to $30°$ and changes at a maximum rate of $\pm 10°/s$. In the rest of this paper β will refer to the collective pitch of the blades.

2.1.6 Wind model

The wind field that hits the turbine is a complex function of the atmospheric conditions, the topology of the landscape and the state and geometry of the turbine. However a much simpler model often suffices. A simple wind model can be described by two parts: a deterministic component and a stochastic component. The deterministic component is described by a function of the mean wind speed at hub height, $v_m(h)$, and the current location of the blade element in question. The stochastic part describes the random deviations from the mean wind speed.

The deterministic part of the wind speed is further divided into two main components: the wind shear caused by the height of the element, i.e. that the wind is slower closer to the ground, and the tower shadow, which refers to the diverting effect of the turbine tower on the wind velocity in the volume below the hub. Both of these effects give rise to dips in the developed torque and force each time a blade moves down and in front of the tower. This leads to oscillatory fatigue on the mechanical subsystem, mainly the drive train and the blades, at three times the frequency of rotation for a three-bladed turbine.

The stochastic part of the wind is random variations in the wind velocity, commonly referred to as turbulence. Since most wind turbines have a rotational frequency several times higher than the turbulence bandwidth, the wind turbulence can be modeled as approximately static each rotation[3]. This means that the turbulence's main effect is to give rise to periodic variations in the torque and force, as the blades move into the same wind speed variations. Since the turbulence is more pronounced the longer out on the blade one looks, it affects the torque more than the force. Thus the aerodynamic torque developed by the entire rotor will have appreciable fluctuations of frequencies around integers of the rotational frequency.

2.1.7 Upstream wind speed profile

The wind turbine extracts energy from the air by slowing it down. Actuator disk theory[3] gives a theoretical lower bound on the velocity at the rotor and far downstream of the rotor, as functions of the wind velocity far upstream, V_∞. This theoretical limit exists because the law of mass preservation implies that for wind to reach the rotor the rotor can not slow the wind down to below a certain value. The mean velocity, V_r, at the rotor plane is defined as

$$v_r = (1-a)v_\infty. \tag{2.5}$$

Here a is a parameter of the wind turbine called the axial induction factor, defined from the thrust, T_{rotor}, on only the rotor

$$T_{rotor} = 2\rho\pi R^2 v_\infty^2 a(1-a), \tag{2.6}$$

and actuator disk theory gives that a is upper bounded by 0.5.

In some cases a more precise model than the above is needed, for example when using LIDAR measurements to predict the mean wind velocity that is going to hit the rotor. One such model for the upstream wind profile can be found through vortex sheet theory[17][7]. It gives the velocity v along the symmetry axis upstream of the actuator disc as a function of the distance from the rotor plane, l, as

$$\frac{v(l)}{v_\infty} = 1 - a[1 - \frac{l}{R}(1 + (\frac{l}{R})^2)^{-1/2}]. \tag{2.7}$$

This function is illustrated in figure 2.4. In [17] both experiments and simulations agree with this model.

2.2 Model Predictive Control

MPC [8] is a control scheme for a dynamic system, that tries to find the input, u_{opt}, that minimizes a cost function, $L(x(t), u(t))$, over a finite time horizon, $[t_0, t_0 + T]$, given some constraints. Here $x(t)$ is the state at time t. Stated more succinctly,

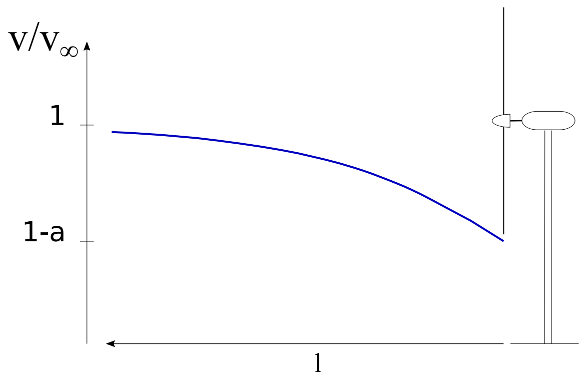

Figure 2.4: Illustration of how vortex sheet theory models the velocity decrease as wind approaches a turbine

$$u_{opt} = \min_{u(\cdot)} \int_{t_0}^{t_0+T} L(x(t), u(t))dt, \tag{2.8}$$

subject to the system dynamics and some bounds on the states, for all $t \in [t_0, t_0 + T]$,

$$\dot{x}(t) = f(x(t), u(t), t), \tag{2.9}$$

$$u_{min} \le u(t) \le u_{max}, \tag{2.10}$$

$$g(x(t), u(t), t) \le 0. \tag{2.11}$$

There is possibly also some constraint on the final state, $x(t_0 + T)$, stated as

$$r(x(t_0 + T)) \le 0. \tag{2.12}$$

This can be applied to a linear system or linearized version of a nonlinear system, giving dynamics

$$\dot{x} = Ax + Bu, \qquad \forall x, u, t. \tag{2.13}$$

Often one wants to drive the states to a reference, but with different costs of deviation from the reference for each state and input. A cost function that incorporates these elements is a quadratic function in the states and inputs,

$$L(x(t), u(t)) = (x - x_{ref})^T Q(x - x_{ref}) + u^T Ru. \tag{2.14}$$

Here Q and R are square, positive-definite matrices with the appropriate dimensions, usually diagonal, representing the cost on the states and controls, respectively.

Sometimes it is also desirable to put a cost on changes to the inputs

$$L(x(t), u(t), \dot{u}(t)) = (x - x_{ref})^T Q(x - x_{ref}) + u^T Ru + \dot{u}^T \Gamma \dot{u}, \tag{2.15}$$

where Γ is a positive-definite matrix with the appropriate dimensions, usually diagonal, representing the cost on the change of the inputs. Note that when this is implemented numerically the derivative is often replace by a first order estimation

$$\dot{u}_i = u_i - u_{i-1}. \tag{2.16}$$

Here the division by the time step has been absorbed into the cost matrix Γ.

Direct single shooting method

The problem in equations 2.8-2.12 can be formulated as a NonLinear Program (NLP), which can be efficiently solved by software libraries like CasADi. To convert the problem from the above formulation to a NLP formulation, it must be discretized in some manner. A method to achieve this is to let the controls be piecewise constant, by dividing up the time horizon, T, into N equal pieces. This gives N parameters for the set of controls, $q_i \in \mathbb{R}^{n_u}$, where n_u is the number of controls and $i = 0, ..., N-1$. The controls are then simply

$$u(t; q) = q_i, \quad \forall t \in [t_i, t_i + 1). \tag{2.17}$$

This gives Nn_u decision variables to be found, concatenated into the decision variable vector q.

The simplest way of proceeding is then to look at the states $x(t)$ for $t \in [t_0, t_0 + T]$ as dependent variables obtained by forward integration of the dynamics given the inputs up to time t, with $x(t_0) = x_0$. This is the so-called direct single shooting method.

The last thing missing is a method of discretizing the state and input constraints. This is done by checking the constraints on a finite set of points along the time horizon, usually the same points where the inputs change values.

The direct single shooting method therefore converts the MPC problem into the following NLP

$$\min_{q \in \mathbb{R}^{Nn_u}} \quad \int_{t_0}^{t_0+T} L(x(t; q), u(t; q))dt, \tag{2.18}$$

subject to

$$x(t_{i+1}) = x(t_i) + \int_{t_i}^{t_{i+1}} f(x(t), q_i)dt, \qquad i = 0, ..., N-1, \tag{2.19}$$

$$h(x(t_i; q), u(t_i; q)) \leq 0, \qquad i = 0, ..., N-1, \tag{2.20}$$

$$r(x(t_0 + T; q)) \leq 0. \tag{2.21}$$

For the system 2.13 with cost function 2.14 this becomes

$$\min_{q \in \mathbb{R}^{Nn_u}} \quad \int_{t_0}^{t_0+T} (x(t; q) - x_{ref})^T Q (x(t; q) - x_{ref}) + u(t; q)^T R u(t; q) dt. \tag{2.22}$$

Direct collocation method

Direct single shooting assumes a piecewise linear state trajectory by only evaluating the constraints at a finite set of points. Another method, called direct collocation, instead approximates the trajectory between each node point by a polynomial. This gives the benefit of allowing us to initialize the state trajectory, typically with information from the previous MPC step, and according to [8] it shows superior local convergence properties in particular for unstable systems.

It divides the time horizon into N collocation intervals. In each collocation interval, $[t_i, t_{i+1}]$, the trajectory is approximated by a polynomial of order j, $p_i(t; v_i)$, with coefficient vector v_i. Similarly the control is parametrized on each interval as a constant or piecewise polynomial $u_i(t; q_i)$, where q_i is the vector containing all the coefficients for each of the input polynomials.

Then j evenly spaced collocation nodes $t_i^{(1)}, ..., t_i^{(j)}$ are chosen in each collocation interval. The state at the start of collocation interval i is denoted s_i. In each collocation interval the following equalities are added to the NLP to enforce the dynamics at the collocation nodes

$$\begin{aligned} s_k &= p_i(t_i; v_i), \\ f(p_i(t_i^{(1)}; v_i), u_i(t_i^{(1)}; q_i)) &= \dot{p}_i(t_i^{(1)}; v), \\ &\vdots \\ f(p_i(t_i^{(j)}; v_i), u_k(t_i^{(j)}; q_i)) &= \dot{p}_k(t_i^{(j)}; v). \end{aligned} \tag{2.23}$$

Continuity across intervals is also enforced by

$$p_k(t_{i+1}; v_i) = s_{i+1}. \tag{2.24}$$

The integrated cost function of each collocation interval is then an analytic function of s_i, v_i, q_i, and is called l_i. State and input constraints are usually enforced at each collocation interval boundary, but could also be done on each collocation node. These constraints, together with the initial conditions and the terminal constraint, become the remaining equations added to the NLP

$$\begin{aligned} s_0 &= x_0, \\ h(s_i, q_i) &\leq 0, \\ r(s_N) &\leq 0. \end{aligned} \tag{2.25}$$

The final NLP is then to minimize the cost function

$$\min_{s,v,q} \quad \sum_{i=0}^{N-1} l_i(s_i, v_i, q_i) + E(s_N), \tag{2.26}$$

subject to equations 2.23, 2.24 and 2.25. This large, but sparse, problem is then solved efficiently with a nonlinear interior point library like IPOPT [32]. Note that the function $E(s_N)$ represent a cost on the final states deviation from a reference, that can be large if it is important that the state ends up close to the reference at the end of the time horizon.

2.3 STAS

STAS is SINTEF's wind turbine simulator, and is explained in detail in [13]. Here follows a short summary of the way STAS is built up.

STAS consists of the interconnection of several different modules, which again consists of the interconnection of submodules. For values of the states, inputs and the parameters percent power output and mean wind speed it gives out the nonlinear dynamics, as well as a high precision linearization of the dynamics. At the highest level STAS is only two modules, a wind turbine module and an electrical grid module. The wind turbine is in turn composed of aeroelastic, electrical, actuator and control modules, while the electrical grid consists of components like cables and transformers. At the lowest level one finds single components, such as blade elements or electrical cables.

Each module gives out their respective nonlinear and linearized dynamics, and these are connected to give the overall nonlinear and linearized dynamics. One such output takes about 40 minutes to compute on a Intel Core i7-8700 CPU running at 3.2 GHz, with 32 GB of RAM. Thus the STAS model is not feasible to simulate in the time domain in the current iteration. What follows now is a brief description of each module in the turbine model.

The aeroelastic module encompasses the local blade aerodynamics and the structural components: blades, low-speed shaft, nacelle, tower, and foundation. The aeroelastic module is further decomposed into an aerodynamic module and a structural module.

The aerodynamic module handles the dynamic relationships between the incoming wind, the turbine wake, blade structural motions, and the aerodynamic forces along the blades. It implements a version of the Blade Element Momentum (BEM) method. The gist of the BEM method is to estimate the change in the flow field due to the vortex wake – not by actually modeling the wake, but rather by assuming an ideal streamwise flow pattern, and performing a control volume analysis, a much-used approach in fluid mechanics.

The structural module accounts for the deformation of the turbine structure, including rotor rotation, under the applied forces. It is a multibody, corotational finite-element beam representation of the wind turbine. The turbine consists of a number of bodies: foundation, tower, nacelle, drive shaft, and blades. Each body can move rigidly in space, and deform elastically.

The electrical module is based on elementary equivalent-circuit models of the turbine and grid electrical components, where the electrical states consist of currents through inductors, and voltages across capacitors.

The actuator module models the pitch actuators. The input is a blade pitch command, and the output from the module is a torque applied to the pitch degree-of-freedom. The actuator dynamics is modeled as a second-order filter, with a smoothed saturation on the pitch angle and pitch rate.

Finally there is a control module, using power set points, measured rotor speed, measured pitch angle, current electrical power, blade root moments and tower velocity to calculate the pitch and power command to the system. This module is, however, turned off in the work done for this thesis, as the goal of this thesis is to develop an alternative to this controller.

STAS has several ways of creating linearizations, and has in this thesis been set to create a reduced linearization with 103 states, where the most relevant simplifications are to only have one mean wind input and to only include the first two modes of each elements moment. The inputs to the full connected system are unitless, as they consists of a combination of effects on the different states, but what will be referred to as input 1 is most directly controlling the power output through controlling the generator speed, while the one that will be referred to as input 2 most directly controls the pitch. There are some other interactions between inputs and states, but these are the most direct couplings.

2.4 Model reduction

The MPC problem requires solving an optimization problem at each time step. How computationally heavy this optimization is depends on how many steps the MPC considers, the convexity of the problem and how many states are involved in the dynamical model used to predict the response of the system. In order to enable the use of MPC as a real time controller of the system, it is therefore beneficial to reduce the number of states. This must of course be done in a way that maintains the accuracy of the model, a goal that can be reformulated as minimizing some norm between the true and reduced system transfer function.

In the following section two different approaches to this end is explored, namely Principal Component Analysis (PCA) [18] and balanced truncation [33].

2.4.1 Principal Component Analysis

PCA is a method for transforming possibly correlated states into an uncorrelated orthogonal basis set, called principle components. By ordering them after singular value size, the most dominant principal components are placed first. To reduce the system is then just a matter of removing the least dominant principal components until the desired order is achieved.

A fixed input is applied to the system, and the state trajectory at certain instances of time is recorded. This gives a so-called matrix of snapshots of the state

$$\chi = [x(t_1),\ x(t_2),\ \ldots,\ x(t_N)] \in \mathbb{R}^{n \times N_s}. \tag{2.27}$$

The number of snapshots, N_s, must be much larger than the number of states, n. This snapshot matrix is then decomposed into a Singular Value Decomposition (SVD) [14], i.e. orthogonal matrices V and W and diagonal matrix S are found such that

$$\chi = VSZ^T, \tag{2.28}$$

where V is $n \times N_s$, while S and Z are $N_s \times N_s$. This can be achieved for any matrix, and is efficiently computed using MATLAB's svd function. The diagonal elements of S are called the singular values of χ. When these elements are put in descending order, a reduced model for the system can be found by using the n_r first columns of V, call it V_r. Approximating the state by

$$x(t) \approx V_r x_r(t), \qquad x_r(t) \in \mathbb{R}^{n_r}, \tag{2.29}$$

gives the reduced dynamics

$$\dot{x}_r(t) = V_r^T f(V_r x_r(t), u(t)), \tag{2.30}$$

$$y_r(t) = h(V_r x_r(t), u(t)). \tag{2.31}$$

In the linear case this is simply

$$\dot{x}_r = A_r x_r + B_r u, \tag{2.32}$$

$$y_r = C_r x_r + Du, \tag{2.33}$$

where $A_r = V_r^T A V_r$, $B_r = V_r^T B$ and $C_r = CV_r$.

2.4.2 Balanced Truncation

In a similar manner to the PCA method, balanced reduction looks at the state trajectory when a certain input is applied. It focuses on an impulse applied to a linear system. This gives the impulse response $h(t) = Ce^{At}B, t \geq 0$. This response can be decomposed into an input-to-state map $x(t) = e^{At}B$ and a state-to-output map $\eta(t) = Ce^{At}$. The so-called controllability Grammian is then

$$\mathbb{P} = \sum_t x(t)x(t)^T = \int_0^\infty e^{At}BBe^{A^Tt}dt, \tag{2.34}$$

and the observability Grammian is

$$\mathbb{Q} = \sum_t \eta(t)^T\eta(t) = \int_0^\infty e^{A^Tt}C^TCe^{At}. \tag{2.35}$$

When the state is linearly transformed into $\hat{x} = Tx$, the two Grammians are transformed by congruence: $\hat{\mathbb{P}} = TPT^T$ and $\hat{\mathbb{Q}} = T^{-T}QT^{-1}$. This means that the eigenvalues of the product $\mathbb{QP}$, $\lambda_i(\mathbb{QP})$ are preserved under the transformation. These eigenvalues are thus input-output invariants of the system, and are what we call the Hankel singular values.

The Hankel singular values are found by first finding the Grammians of the system as the solution to the Lyapunov equations

$$A\mathbb{P} + \mathbb{P}A^T + BB^T = 0, \tag{2.36}$$

$$A^T\mathbb{Q} + \mathbb{Q}A + CC^T = 0. \tag{2.37}$$

The Hankel singular values are then the square roots of the eigenvalues of the product $\mathbb{PQ}$. An upper triangular matrix U, satisfying $\mathbb{P} = UU^T$ and a lower triangular matrix L, satisfying $\mathbb{Q} = LL^T$ are then found. Doing a SVD of $U^TL = Z\Sigma Y^T$ gives a balanced transformation matrix

$$T_b = \Sigma Z^T U^{-1} = \Sigma^{-\frac{1}{2}} Y^T L^T, \tag{2.38}$$

with inverse

$$T_b^{-1} = UZ\Sigma^{-\frac{1}{2}} = L^{-1}Y\Sigma^{\frac{1}{2}}. \tag{2.39}$$

When the system is transformed with T_b, giving $A_b = T_bAT_b^{-1}$, $B_b = T_bB$ and $C_b = CT_b^{-1}$ the solution to the two Lyapunov equations become equal and diagonal:

$$\mathbb{P} = \mathbb{Q} = \Sigma = diag(\sigma_1, \dots, \sigma_n). \tag{2.40}$$

If the system is then ordered so the Hankel singular values are descending, a model for the system reduced to $n_r < n$ states is achieved by simply truncating the balanced system at the desired degree, i.e. by taking the $n_r \times n_r$, $n_r \times m$, $p \times n_r$ leading blocks of A_b, B_b and C_b, respectively.

The benefit to this reduction is that it can be shown [11] to satisfy the following bounds on the H_∞-norm

$$\sigma_{n_r} \leq ||\Sigma - \hat{\Sigma}||_\infty \leq 2(\sigma_{n_r+1} + \sigma_{n_r+2} + \ldots + \sigma_n). \tag{2.41}$$

The H_∞-norm is defined as the maximum of the highest peak of the frequency response, i.e.

$$H_\infty = \sigma_{max}[D + C(j\omega - A)^{-1}B]. \tag{2.42}$$

2.5 Interpolation of Linear Time-Invariant systems

When a LTI system is acquired from a nonlinear system it only gives a good approximation for the system dynamics when the state is close to the linearization point. For highly nonlinear models this will generally not be adequate. To remedy this while allowing for the continued use of linear control algorithms, the system could be linearized for many state values, spaced finely throughout the state space. These linearizations could then be interpolated to give a mapping from state to linear system to be acquired at the start of each control loop. This will only capture the systems dependency on the states, not any possible dependency on any of the derivatives, $\dot{x}, \ddot{x}, ...$, but in most cases this will be sufficient.

For large dimensionality this is unfortunately often times not feasible, as it would require the acquisition of a lot of linearized systems, which can be both expensive and time-consuming. However, in many cases it is possible to parametrize the set of equilibrium points by a number of parameters that are fewer than the number of states. This leads to a LPV system, which is the industry standard way of efficiently acquiring a linearized system from a nonlinear model that remains accurate for the whole state space.

2.5.1 Coherence of LTI systems

When creating a LPV system by interpolation of many LTI systems, a very important first step is to make sure the linearized systems for different parameters are coherent and transform them in such a way that the states represent the same physical state in each system. Coherence refers to the fact that such a transformation is possible to find. More accurately, coherence is defined in [5] in the following way: For ease of writing, a linear system

$$\dot{x} = Ax + Bu \tag{2.43}$$
$$y = Cx + Du \tag{2.44}$$

is referred to as H, and written as

$$H = \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right]. \tag{2.45}$$

Now, say one has a set of k LTI models

$$H_l = \left[\begin{array}{c|c} A_l & B_l \\ \hline C_l & D_l \end{array}\right] \tag{2.46}$$

obtained for fixed operating conditions $p = \tilde{p}_l$, $l = 1, \ldots, k$, from a parameter dependent nonlinear system. The LTI models are said to be represented in a coherent state space form if their system matrices A_l, B_l, C_l and D_l, for $l = 1, \ldots, k$, are evaluations of

$$H(p, \dot{p}, ...) = \left[\begin{array}{c|c} A_l & B_l \\ \hline C_l & D_l \end{array}\right] = \left[\begin{array}{c|c} T(\tilde{p})A_S(\tilde{p})T(\tilde{p})^{-1} & T(\tilde{p})B_S(\tilde{p}) \\ \hline C_S(\tilde{p})T(\tilde{p})^{-1} & D_S(\tilde{p}) \end{array}\right] \tag{2.47}$$

where $T(p)$ is a bounded non-singular continuously differentiable transformation matrix, depending on the same parameters p that the equilibrium states depend on. The matrices $A_S(p)$, $B_S(p)$, $C_S(p)$ and $D_S(p)$ are the true linear parts of the system dynamics at the parameter value p. Coherence is usually a problem only in two scenarios. The first is when the local linearized model is found empirically by fitting to the input-output map of the system. If the LTI systems are linearizations of the nonlinear dynamics, they should be coherent. The other scenario is when the LTI systems are acquired through such a linearization of the nonlinear dynamics, but then reduced to a smaller state space to speed up computation time. The reduced systems are then unlikely to be coherent unless some thought was put into the reduction.

Most methods for checking coherence also give rise to the solution to the next problem: transforming each local linear system so they represent the states in the same way. A method of checking coherence and finding this transformation is given in [5].

2.5.2 Transforming the reduced systems to a common basis

When LTI systems are reduced to a smaller state space in order to speed up computations, their reduced systems are typically not reduced in the same manner, meaning a set of coherent LTI systems with the same basis will have reduced systems that do not represent the states in the same way. To remedy this one can use a single projection matrix to reduce all the systems, but this often leads to poor performance in parts of the state space where the projection hides some of the important dynamics. A better solution is to transform the reduced systems into having the same basis. This is done in [21], and an outline of the method is now given.

Several coherent LTI systems are found for different parameter values p_l, $l = 1, \dots k$:

$$\begin{aligned} \dot{x}(t) &= A_l x(t) + B_l u(t), && (2.48) \\ y(t) &= C_l x(t). && (2.49) \end{aligned}$$

Each of the k models is reduced to n_r states using different projection matrices $V_l, W_l \in \mathbb{R}^{n \times n_r}$. This leads to k reduced order systems

$$\begin{aligned} \dot{x}^*_{r,l}(t) &= W_l^T A_l V_l x_{r,l}(t) + W_l^T B_l u(t) = A_{r,l} x_{r,l}(t) + B_{r,l} u(t), && (2.50) \\ y_{r,l}(t) &= C_l V_l x_{r,l}(t) \qquad\qquad = C_{r,l} x_{r,l}. && (2.51) \end{aligned}$$

The LPV system is then found as

$$\begin{aligned} \dot{x}_r(t) &= A_r x_r(t) + B_r u(t), && (2.52) \\ y_r(t) &= C_r x_r(t). && (2.53) \end{aligned}$$

where

$$\begin{aligned} A_r &= \sum_{l=1}^{k} \omega_l(p) A_{r,l}, \qquad B_r = \sum_{l=1}^{k} \omega_l(p) B_{r,l}, && (2.54) \\ C_r &= \sum_{l=1}^{k} \omega_l(p) C_{r,l} && (2.55) \end{aligned}$$

interpolate the reduced order system matrices. Without transforming the reduced systems in some manner this will however not give good results, as the reduced systems have different bases and interpolating them might cause odd dynamics. The reader is referred to [21] for an example of this problem.

Each reduced system can be transformed in the following manner without changing the input-output behavior

$$\begin{aligned} \dot{x}_{r,l}(t) &= M_l A_r T_l^{-1} x^*_{r,l}(t) + M_i B_l u(t) = A^*_{r,l} x^*_{r,l}(t) + B^*_{r,l} u(t), && (2.56) \\ y_{r,l}(t) &= C_l T_l^{-1} x^*_{r,l}(t) \qquad\qquad = C^*_{r,l} x^*_{r,l}. && (2.57) \end{aligned}$$

The method seeks to make the reduced systems represent the states in the same manner by finding M_l's and T_l's that do so.

It starts by finding the T_l's that reproject all the reduced states into a common subspace. This does however not make the states $x^*_{r,l}$ represent the same thing, since when they are projected back into the original full state space by $\hat{x} = V_l x_{r,l} = V_l T_l^{-1} x^*_{r,l}$ they will still lie in the subspace spanned by their respective projection matrices V_l, and thus not have the same meaning. To remedy this, the T_l's are chosen so they make the state vectors $x^*_{r,l}$ compatible with respect to a subspace spanned by the columns of a matrix $K \in \mathbb{R}^{n \times q}$.

Two state vectors $x^*_{r,l} \in \mathbb{R}^{q \times n_r}$ are called compatible w.r.t a matrix $K \in \mathbb{R}^{n \times n_r}$ if the images of their basis vectors under a transformation $T_l \in \mathbb{R}^{n_r \times n_r}$, backprojecting using the matrices $V_l \in \mathbb{R}^{n \times n_r}$ and reprojecting into the subspace spanned by the columns of matrix K, are identical.

The result of choosing the T_l's in this manner is that starting from a given reduced state vector $x^*_{r,l}$, transforming it to a local reduced coordinate system

$$x_{r,l} = T_l^{-1} x^*_{r,l}, \tag{2.58}$$

projecting it back to the original subspace using the associated projection matrix V_l,

$$\hat{x}_l = V_l x_{r,l} = V_l T_l^{-1} x^*_{r,l}, \tag{2.59}$$

and finally reprojecting it to the subspace spanned by the columns of K,

$$K^T \hat{x}_l = K^T V_l x_{r,l} = K^T V_l T_l^{-1} x^*_{r,l}, \tag{2.60}$$

the same vector is obtained for all the reduced systems. The T_l's that achieve this is

$$T_l = K^T V_l, \tag{2.61}$$

leading to

$$K^T V_1 x_{r,1} = K^T V_2 x_{r,2} = \ldots = K^T V_k x_{r,k} =: x^*_r. \tag{2.62}$$

Note that K needs to be orthogonal in order for this to work, since it needs to be possible to project the common state vectors x^*_r back to the original high dimensional state space by $K x^*_r$. It also needs to be chosen such that no transformation matrices T_l become singular.

To choose the K matrix, the method tries to find the n_r directions in the state space that are most important to approximate the dominant dynamics of the involved local models, where n_r is the order of the reduced models. Call the matrix of all the columns of all the V_l's for V_{all},

$$V_{all} = [V_1, V_2, \dots, V_k]. \tag{2.63}$$

Taking the SVD of this gives

$$V_{all} = U \Sigma N^T, \tag{2.64}$$

where the first $k \times n_r$ columns of the orthogonal matrix U form a basis for the subspace spanned by V_{all}. Sorting the SVD by the size of the singular values means the basis vector is sorted by their relative importance, so taking K to be the first n_r columns of U means K captures the most important directions in V_{all}.

Transforming the systems with the transformation matrices T_l therefore means all the reduced systems will backproject the same reduced state to the same state $\hat{x}$, but still does not guarantee that the state equations are given with respect to the same basis. The reduced system dynamics

$$\dot{x}_r(t) = W^t A V x_r(t) + W^T B u(t), \tag{2.65}$$

lies in the orthogonal subspace of W. If there was a full column rank matrix $S \in \mathbb{R}^{n \times n_r}$ with the property that

$$det(W^T S) \neq 0, \tag{2.66}$$

projecting the system dynamics in the following manner

$$(W^T S)^{-1} \dot{x}_r(t) = (W^T S)^{-1} W^T A V X_r(t) + (W^T S)^{-1} W^T B u(t) \tag{2.67}$$

would map vectors along the orthogonal complement of W onto the subspace spanned by S. This means that the reduced system's dynamics would now be rooted in the subspace spanned by S, without changing the solution $x_r(t)$.

Since a full column rank matrix with the above needed property 2.66, K, has already been found, a reasonable choice for the M_l's is

$$M_l := (W_l^T K)^{-1}. \tag{2.68}$$

This, in summary, transforms the state so all the systems backproject the same vector into the same full state vector and gives all the reduced systems a common basis. The method thus makes it possible to transform the reduced systems in such a way that they can be interpolated.

2.5.3 Independent interpolation of elements

When coherence has been confirmed and the LTI systems are transformed into the same basis, the LPV system is obtained by interpolation of the LTI systems. The standard way of doing this is to assume that the elements in the system matrices are independent of each other and only functions of the parameters p. The problem is then to find the best fit for each element, that captures the non-linearity without making the LPV too computationally expensive. Typically one tries to fit a polynomial or some other basis function on the element, increasing the order of the interpolating function until a trade-off is achieved between computation time and accuracy. This leads to a mapping between parameters p and linear system, i.e.

$$H(p) = \left[\begin{array}{c|c} A(p) & B(p) \\ \hline C(p) & D(p) \end{array}\right]. \tag{2.69}$$

This mapping can then be used to get a linear system at each control step, allowing for the use of linear control methods of a nonlinear system if the control step is small enough and the interpolation is precise. What step size is small enough and how to measure a good interpolation is of course non-trivial.

Chapter 3

Method

In this chapter the development of the system is explained. First it covers the implementation of the linear MPC, then how the system was reduced to speed up computations, before the attempts at reducing the change of moments of the turbine are explained. It then goes on to talk about feed forward MPC using LIDAR measurements and explains the method used to create a LIDAR noise model to test the performance of the feed forward MPC. Finally it outlines how the MPC was modified to control a LPV system instead of a LTI system.

For all the different iterations of the MPC the pitch is restricted to the interval $[-2, 30]$ degrees ($\approx [-0.035, 0.524]$ radians). In all scenarios the MPC runs an optimization over the time horizon at each step, but then only the first of the calculated inputs is used. At the next iteration the optimization is done again with the starting state being the latest state from the simulation run in parallel with the MPC. In all situations the simulated system with controller is allowed to converge before any changes in wind is applied.

As explained in section 2.3, the inputs are defined as unitless in this thesis, but input 1 controls the power set point through controlling the generator speed and voltages, while input 2 controls the motors controlling the collective pitch of the blades.

3.1 Linear MPC

The first step towards a MPC for the STAS model was to make a MPC controlling the power of a single linearization of the model. Using the CasADI [1] framework, the collocation NLP was set up, and solved via the nlpsolver function, using the IPOPT [32] solver. The first implementation had the MPC controlling the generator speed and pitch reference in a stepwise manner. The resulting NLP is equations 2.23-2.26, with the instantaneous cost

$$l_i(s_i, v_i, q_i) = (x_{ref} - s_i)^T Q (x_{ref} - s_i), \tag{3.1}$$

where s_i, as a reminder, is the state at the ith collocation boundary. Q is all zero except for the diagonal elements corresponding to the power state. Note that the STAS model has a power state that is most dependent on the difference between generator and rotor rotational speed, some voltages and currents in the generator and grid, as well as on the blade pitch angles and the wind input. The reference is zero except for the power state, which the system is told to drive towards the rated speed of 10 MW.

The stepwise input lead to the addition of harmonics to the system, so in an attempt at reducing this addition of harmonics the NLP was set up with linearly changing inputs over each collocation interval. That is to say, the MPC controls the value of the inputs at the collocation boundaries, while in between it moves linearly between the boundary input values. To state it mathematically, if t is in collocation interval i, giving $t \in [t_i, t_{i+1})$, then

$$u(t) = u_i + \frac{t - t_i}{t_{i+1} - t_i}(u_{i+1} - u_i). \tag{3.2}$$

Another approach that was tested to remove the harmonics, was to add two lowpass filter states, making the MPC control the inputs to this filter, and having the filter outputs be the inputs to the turbine. Several tunings for this was tried before good results were achieved. The first attempt at lowpassing the system inputs was with stepwise input, but linearly changing inputs to the filter was also tried. With the sample and hold inputs the gains for the power input filter gave best performance when set to 0.01, while the best gain for the pitch input filter was found to be 0.1. When using linearly varying inputs to the filter it was found that the gains could be set a little higher without adding harmonics, but it did not lead to as good power tracking than with the stepwise input to the filter. In this case the power filter gain was set to 1 and the pitch filter gain set to 0.3.

The inclusion of the lowpass filter states gave better results, but also made the system more computationally heavy, and it was no longer possible to keep the MPC running in real time without removing all but two of the control intervals, which gave very poor performance. In addition, the author believed that having the MPC control set points of a PI controller might lead to better performance, but this would mean increasing the computation time even more as it meant adding two more states. Therefore the next step was to reduce the system to fewer states.

3.2 Model reduction

To make the system run in real time the linearized system had to be reduced to fewer states. Real time in this thesis refers to being able to finish calculating the optimal input to the system before the next input is demanded. If the time horizon of the MPC is T seconds divided into N intervals, then this definition of real time capability implies that the optimal input should always be found in less than T/N seconds.

To achieve this the system was first reduced with balanced reduction using the tbr function of the sssMOR [6] toolbox, but it could not be reduced to any less than 62 states without loss of convergence to the desired power. This setup had the MPC control a lowpassed input, and two of the states where therefore the internal states of this filter.

The system was then reduced with PCA, by simulating the system with constant input for 120 seconds, at step size of 0.01, giving 12000 snapshots. Every 10th of these were collected into a snapshot matrix containing 1200 snapshots. An ordered SVD transformation of this matrix was made using MATLAB's svd function. Taking only the twenty first columns of the V matrix in the svd(see section 2.4.1) gave a projection matrix used to project the system down to only 20 states, which allowed the MPC to run in real time with as much as 20 control intervals on a 40 second time horizon.

The result of this was a system reduced from 103 states down to only 22 states, since two states was added back in to serve as the states of the filter used to lowpass the input. The resulting controller ended up converging fast, but with a small bias. This is a typical case where including a PI filter would give better performance, so this was implemented next.

3.2.1 Controlling PI set points

To remove the bias stemming from the error in the reduction, the MPC was made to control the set points of a PI controller controlling power and pitch. The time horizon was 25 seconds with 5 control intervals, which gave a run time with a mean of 0.35 seconds and a max of 0.548 seconds, meaning it only used 33.33 seconds of CPU time for a simulation lasting 500 seconds. This leaves room for more sophisticated methods, like feed forward wind measurements, and extending the prediction model to a LPV system.

3.3 Reducing moment derivatives

To reduce fatigue, it is desirable to reduce the change in the moments of the different parts of the wind turbine. An attempt at making the MPC achieve this was done by putting a cost on the derivative of the moments of the blade roots and the tower nodes. This did however only lead to a slightly faster convergence to zero of the moment derivatives, but made the power tracking much worse. No further attempts at reducing the fatigue was therefore undertaken.

3.4 LIDAR feed forward

With modern LIDAR or RAdio Detection And Ranging (RADAR) sensors, it is possible to use the reflection of electromagnetic waves off small particles in the air to remotely estimate the wind speed. There are now several companies specializing in such sensors for wind turbines. Especially LIDAR sensors have become popular.

According to [25] simulations show that the Root Mean Square (RMS) error between measured and actual wind speed is 0.26 $\mathrm{m\,s^{-1}}$ below 50 m from the rotor and 0.62 $\mathrm{m\,s^{-1}}$ at 275 m from the rotor. In [19] the deviation between a LIDAR system and a laser Doppler

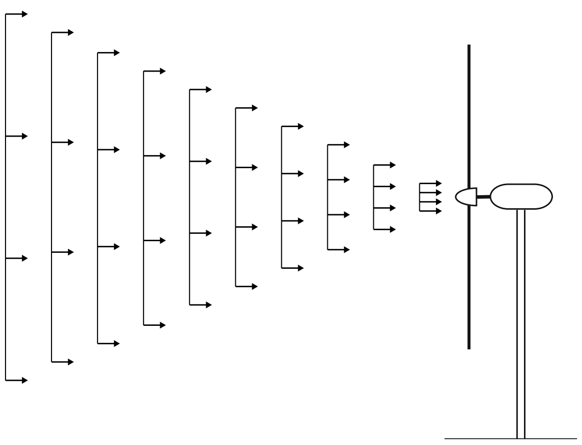

Figure 3.1: Side view of a simplified version of the wind measurement field from a hub mounted LIDAR with 10 measurement planes. The fact that each measurement plane consists of several vertical slices is not possible to see on the drawing as each vertical slice is situated behind each other in this side view.

anemometer was found to be below 0.1 %. According to [4], using state of the art LIDARs, it is possible to get an estimate of the wind speed that is within 0.5 percent of the wind speed measured with a mast top-mounted cup anemometer at 2.5 rotor diameters upstream of the turbine. The DTU 10 MW reference turbine considered in this thesis has a rotor diameter of 178.3m m, which would mean this error would be at 445.75 m, which is 45.75 m further than the LIDAR in this thesis measures. So it therefore seems likely that the LIDAR will provide some upstream wind velocity information, and thus be beneficial.

Following the specs of the ZX-TM, a state of the art hub mounted LIDAR from ZX LIDARS [34], such a sensor gives 10 measurement planes equally spaced between 10 m and 400 m away from the sensor. The measurement planes consists of measurements of the component of the wind speed moving towards the sensor, divided into 10 vertical slices with 30 point measurements in each slice. A simplified side view of this is illustrated in figure 3.1. Placing this sensor at the nacelle, pointed out between the blades, gives 10 measurement planes aligned with the turbine's plane of rotation. It measures at 50 Hz, a rate much higher than how often the MPC will be updated, so it is assumed that the MPC has fresh measurements each iteration.

Because of how STAS simplifies the actual wind field at the rotor plane into a single mean rotor plane orthogonal velocity, the measurement planes from the LIDAR are only used for their mean value. These measurements represent an estimate of the mean of the orthogonal part of the wind speed, $\hat{v}(l)$, as a function of distance from the rotor plane, l, sampled at the measurement distances $\mathbf{l} = [l_1, \ldots, l_k]$.

To make this information usable for the MPC, it is necessary to convert this to an estimate of the mean wind speed at the plane of rotation as a function of time, $\hat{v}(t)$. The κ measurements $\hat{v}_{j,space} = \hat{v}(l_j), \forall j = 1, \ldots, \kappa$, could give rise to κ estimates $\hat{v}_{j,time} = \hat{v}(t_j)$ of the wind speed at the nacelle at times $\mathbf{t} = [t_1, \ldots, t_\kappa]$, if one could somehow estimate how the speed of the measured wind planes would change as it moves towards the nacelle and also when they would arrive at the nacelle. The latter of these two is rather easy if the former estimate is good.

To reformulate the latter question: If one knows the starting position, $l_{particle,0}$, of a particle on a line and what the speed, $v_{particle}(t)$, of the particle along the line will be for all times, how does one find the time, t_{impact}, where it hits $l = 0$? This is equivalent to setting the origin of the position axis at the starting position of the particle, and finding when the particle arrives at $l = -l_{particle,0}$. This leads to the following equation

$$l_{particle,0} = -\int_0^{t_{impact}} v_{particle}(t)dt. \tag{3.3}$$

Solving this equation for t_{impact} will then give the answer.

If the effect of the blades on the wind is neglected a rather simple, yet efficient approximation for the evolution of the speed of this particle would be to assume that the speed remains constant. This would simplify equation 3.3 to

$$l_{particle,0} = -\int_0^{t} v_{particle}(0)dt = -v_{particle}(0)t_{impact}, \tag{3.4}$$

which leads to the following expression for the time when the particle hits the turbines

$$t_{impact} = \frac{-l_{particle,0}}{v_{particle}(0)}. \tag{3.5}$$

This would mean that the mean wind velocity at the rotor plane at time t_{impact} could be approximated as $v_{particle}(0)$. In other words, if the velocity of each particle does not change as it approaches the turbine, the transformation between measured velocities in space, $v_{\kappa,space}$, at time t_{meas}, and estimated velocities at the nacelle at time t, is the transformation $t_\kappa = t_{meas} - l_\kappa / v_{\kappa,space}$. Note here that the positive x-axis is pointing out from the hub, so assuming the turbine is oriented against the wind, $v_{\kappa,space}$ is always negative.

Of course in reality the velocity of each particle does not remain constant. The assumption about constant velocity could still be argued to be the best estimate one has if the effect of the turbine blades was negligible and no more information was given. The effect of the turbine blades does, however, need to be accounted for, as a wind turbine extracts energy from the wind by shedding velocity from it.

To account for the effect of the turbine on the wind speed measurements, the upstream wind speed from vortex sheet theory could be used. Note that none of the simulations in this thesis include this effect. This is because any such effect would simply be removed again in the controller, and therefore not give any indication of the whether the system for accounting for it works. However, in a real controller this effect would need to be accounted for, and a method for doing so is therefore now included for completeness.

The upstream wind velocity can be approximated by equation 2.7. In [17] some error is observed between the experimental results and this model, but it is rather small, in the order of 2-4 % in the $l/D = -2$ range. This error should be dominated by the measurement noise from the LIDAR.

This model for how the wind velocity changes as it approaches the rotor could then be used to predict the mean wind speeds that will hit the rotor blades. To achieve this, one must find the velocity that the measured wind plane will have when it hits the rotor, and at what time it hits the rotor. First the different v_∞ for each measurement is found by solving 2.7 for V_∞

$$v_\infty = \frac{V(l_{meas})}{1 - a[1 - \frac{l_{meas}}{R}(1 + (\frac{l_{meas}}{R})^2)^{-1/2}]}. \tag{3.6}$$

This can then used to find the velocity at the time of impact

$$v(x = 0) = v_\infty(1 - a), \tag{3.7}$$

and the velocity of the measured wind plane at all velocities l as

$$v(l) = (1 - a[1 - \frac{l}{R}(1 + (\frac{l}{R})^2)^{-1/2}])v_\infty. \tag{3.8}$$

The time of impact of each of these measured wind planes can then found by integrating using Euler's method

$$x_{\kappa+1} = l_\kappa + v(l_\kappa)dt \tag{3.9}$$
$$t_{\kappa+1} = t_\kappa + dt \tag{3.10}$$
$$t_0 = t_{meas} \tag{3.11}$$
$$l_0 = l_{meas} \tag{3.12}$$

until $l_{\kappa+1} \leq 0$. The time of impact is then approximated as $t_{impact} \approx t_{\kappa+1}$, which will give a good approximation if dt is sufficiently small.

Using the ZX-TM LIDAR and equation 3.5 gives predicted mean velocities, $\hat{v}_{i,time}, i \in [1, 10]$, at the rotor plane for 10 future time points, $\mathbf{t} = [t_1, ..., t_{10}]$. These are then smoothed using a lowpass filter, making sure that the speed at the nacelle at the measurement time is unchanged. Then these velocities are interpolated using a second-order spline preserving the first derivatives. This velocity as a function of time is then used in the prediction step of the MPC.

3.4.1 Measurement noise and filtering

Zero mean Gaussian noise was then added to each measurement, with the power of the noise increasing linearly as a function of distance from the turbine to account for the fact that measurements should give less information about future wind inputs the further away they are. To combat this increased noise, the predicted wind inputs stemming from the measurements where filtered with a lowpass filter with cutoff frequency of 0.1. All the resulting smoothed wind speed predictions where then given a correction to make the current wind speed match the last applied wind input. The reason for not adding any noise on the current rotor plane average velocity estimate is that any error in this estimate will affect the PI controller as well as the MPC, since the estimated power output of the turbine is a function of the same estimate of the rotor plane average velocity. On many turbines this estimate is made by a wind observer using the measured generator and rotor speed together with the measured pitch, by inversion of a static aerodynamic model [20].

3.4.2 Finite difference noise model

In order to more robustly test the performance of the feed forward MPC a more realistic noise model is needed. The main problem with simply adding white noise on each measurement at each step is that it neglects some of the spatial and temporal correlation that would be seen in real LIDAR measurements. The solution that ended up being implemented used a finite difference scheme to solve the transport equation

$$\frac{\partial \alpha}{\partial t} + c\frac{\partial \alpha}{\partial x} = 0, \tag{3.13}$$

with some modifications that will be explained below.

The main idea is to discretize the space 800 m in front of and behind the wind turbine into 200 states, $\alpha_i,\ i \in [1, 200]$, representing the mean velocity of the wind plane at each of the distances and then getting an approximation of the rate of change of each of these velocities through a finite difference approximation of equation 3.13. At first a first order centered approximation was tried, which has the following expression

$$\frac{\partial \alpha_i}{\partial x} \approx \frac{\alpha_{i+1} - \alpha_{i-1}}{2d}, \tag{3.14}$$

where d is the distance between the states. Plugging this into eq. 3.13 and rearranging gives the following approximation of each of the states derivative

$$\frac{\partial \alpha_i}{\partial t} \approx \frac{\alpha_{i-1} - \alpha_{i+1}}{2d}. \tag{3.15}$$

This gives a state space representation

$$\dot{\alpha} = \frac{1}{2d}\begin{bmatrix} 0 & 1 & 0 & 0 & \dots & 0 & 0 \\ -1 & 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & -1 & 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & -1 & 0 & 1 \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 \end{bmatrix} \alpha + \frac{-1}{2d}\begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ \vdots \\ 0 \\ 0 \end{bmatrix} w, \tag{3.16}$$

where w is the inlet velocity driving the changes to the rest of the states. The state furthest downwind of the rotor is set to have zero derivative, so it remains at the starting value of 14 $\mathrm{m\,s^{-1}}$. Because of the first order approximation the system is unstable when time simulated with Euler integration, leading to some very unphysical wind velocities of above 100 $\mathrm{m\,s^{-1}}$ in some parts of the state space. A second order forward difference,

$$\frac{\partial \alpha_i}{\partial x} \approx \frac{-\alpha_{i+2} + 4\alpha_{i+1} - 3\alpha_i}{2d}, \tag{3.17}$$

was therefore used instead, giving the following approximation of the state derivative

$$\frac{\partial \alpha_i}{\partial t} \approx \frac{\alpha_{i+2} - 4\alpha_{i+1} + 3\alpha_i}{2d}. \tag{3.18}$$

This produces the state space representation

$$\dot{\alpha} = \frac{1}{2d}\begin{bmatrix} -4 & 1 & 0 & 0 & \dots & 0 & 0 \\ 3 & -4 & 1 & 0 & \dots & 0 & 0 \\ 0 & 3 & -4 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 3 & -4 & 1 \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 \end{bmatrix} \alpha + \frac{-1}{2d}\begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ \vdots \\ 0 \\ 0 \end{bmatrix} w. \tag{3.19}$$

Time integrating this equation with first order Euler integration with only a step input as the inlet velocity w gives the state velocities as a function of time and distance from rotor seen in figure 3.2. Note that the figure only shows the states located between 0 and 600 meter upstream of the rotor. The velocity at zero distance from the rotor is the velocity at the rotor plane that is used for all subsequent simulations as the actual rotor plane average wind velocity. All inlet noise, process noise and gusts are added to a separate velocity state space obeying the same dynamics in eq. 3.19, that is sampled at the measurement distances to get the LIDAR measurements. The rationale is that adding inlet noise to this measured velocity state space will give correlated noise propagating towards the rotor, while process noise will simulate the uncorrelated variation in the measured wind speed. The added gusts are thought to further test the robustness of the controller, possibly thought of as a

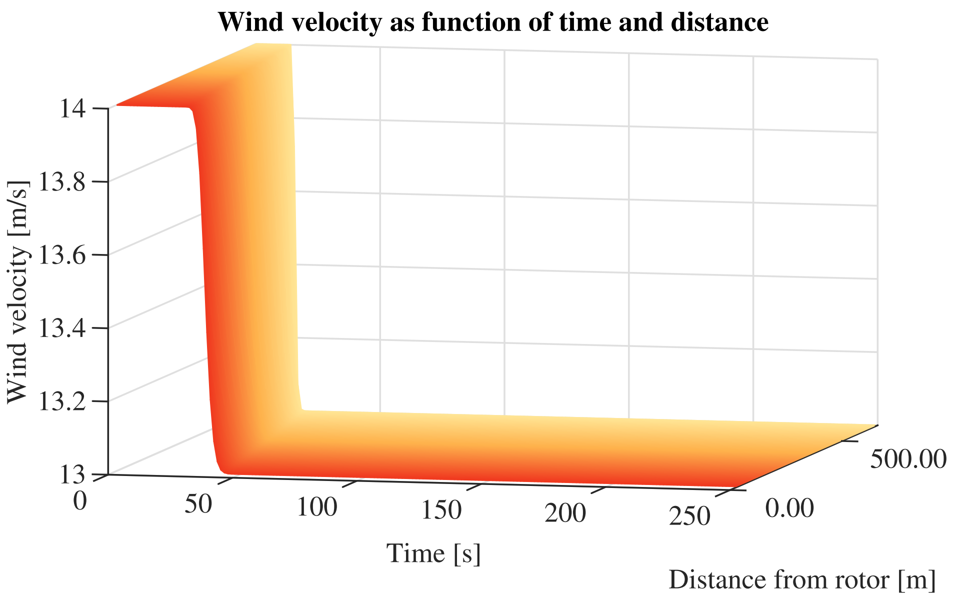

Figure 3.2: Wind velocity as a function of distance from rotor and time, without any noise. The input to the simulation is given by the velocity at zero distance from the rotor.

gust moving into the LIDAR measured volume but then not hitting the rotor plane. How these disturbances are added is now explained.

Noise is added on the inlet by adding zero mean Gaussian noise to the inlet velocity. Process noise is added by adding uncorrelated zero mean white noise to all the state derivatives. What will be referred to as gust is inserted by giving a temporary increase to the derivative of a state upstream of the rotor. In the most extreme case depicted in the results the Gaussian inlet noise has a power of 2, the process noise has a power of 0.5 while the gust is an increase of 10 $\mathrm{m\,s^{-2}}$ to the derivative of the state at 350 m upstream, lasting for 4 seconds. The velocity state space that the LIDAR measures from is plotted in figure 4.15.

3.5 Linear Parameter-Varying System

Since the full wind turbine system is highly nonlinear, the LTI system acquired by linearizing the STAS model will only be valid in a small region around the linearization point. A LPV system is created by interpolating many of these linearization points to account for the non-linearity. Since the steady state equilibrium points of STAS only depends on two parameters, the mean wind input to the system and the fraction of actual power output to the rater power, this process does not need to linearize for all state values, only for the parameters.

The full system was linearized for all combinations of ten different values for the two parameters, spanning the whole range of possible parameter values. This gave a grid of

100 linear systems, equally spaced throughout the parameter space. Coherence between the systems is assumed, since these systems came from high precision linearizations of the nonlinear model. Thus the process of obtaining the interpolated LPV system was now just a matter of interpolation. The linearization process took three days each time, but after many attempts where some bug caused the structure of the linearized systems to become unphysical for some parameter values, the project was abandoned due to time constraints. A smaller set of three verified linearizations was provided by SINTEF, linearized at mean wind speeds 14 $\mathrm{m\,s^{-1}}$, 12 $\mathrm{m\,s^{-1}}$ and 8 $\mathrm{m\,s^{-1}}$, and it was decided to use these instead. This lead to a less-than-ideal interpolated system, but at least it allowed further investigations into whether the main ideas worked.

Following the standard approach in the literature, each element in each of the system matrices was assumed to only need to be fitted with the same element from the other linearization points. The original idea was to choose the order of the interpolating polynomials by finding a tradeoff between computation time, memory use and some error norm on the transfer function. But since the LPV ended up being constructed from only three linearizations, all at the same percent power output, the interpolating polynomial ended up being of order three, interpolating the system as a function of the mean wind input.

This gave a mapping between parameters and LTI systems for all possible parameter values. At the start of each MPC iteration, the current parameters are used to give the current LTI system, which is then used by the MPC to predict the system behavior. To allow a similar performance and computation time as for the single linearization, this system also needs to be reduced. The first reduction attempt was with a single projection matrix since an interpolation of several reduced systems does not work unless the reduced systems are transformed in some manner first. The optimal projection matrix was found by creating and concatenating the snapshot matrices for all linearized systems and using this to create the PCA based projection matrix as explained in 2.4.1. This was then applied at each step to the current linearized system, to give a reduced system. This reduced system was then used by the MPC to control the set points of the PI regulator, like before. This did however not lead to satisfactory power tracking. Adding a cost on the rate of change of the inputs made the power tracking made the system stable, but the power tracking was still very bad, so a better solution was sought after.

Instead of using only one static reduction, each system was reduced and transformed according to the discourse in section 2.5.2. The reduced systems could then be interpolated, which was done in a similar manner to the full system interpolation.

Chapter 4

Results

This chapter details the results that came out of the work done for the thesis. It first shows the response of the different flavors of the full linear MPC, before the results of the model reduction are presented. Then the response of the feed forward MPC is displayed, including the performance of feed forward MPC on the finite difference noise model. After this the performance of different controllers on the interpolated LPV system is demonstrated, before ending with the MPCs performance with regards to fatigue reduction.

In all the tested systems and scenarios the tuning of the PI controller is the same. This is deliberately done so this does not cause any unfair advantage to any of the controllers. Furthermore, in the full linear case, the scenarios testing the linear reduced systems, as well as the scenarios detailing the attempts at reducing the moment derivatives, the MPC has a time horizon of 25 seconds divided into 5 control intervals.

4.1 Full linear system

In this section the results of different variations of the full linear MPC can be found.

The MPC predicting using the full linear system, controlling a stepwise input, gave the results seen in figure 4.1. The power tracking is poor, and contains a lot of jumps as the system does not respond well to jumps in the input.

Running the MPC on the full linear system with the MPC controlling a filtered input gave the plots seen in figure 4.2.

Running the MPC on the full linear system with linearly changing inputs gave the plots seen in figure 4.3. This lead to worse power tracking capabilities than the lowpassed input, but less jitter on the system. A hybrid version was therefore implemented, with the MPC controlling linearly varying inputs to a filter, the output of which is used as inputs to the system. This lead to the performance seen in figure 4.4. Lowpassing the input looked

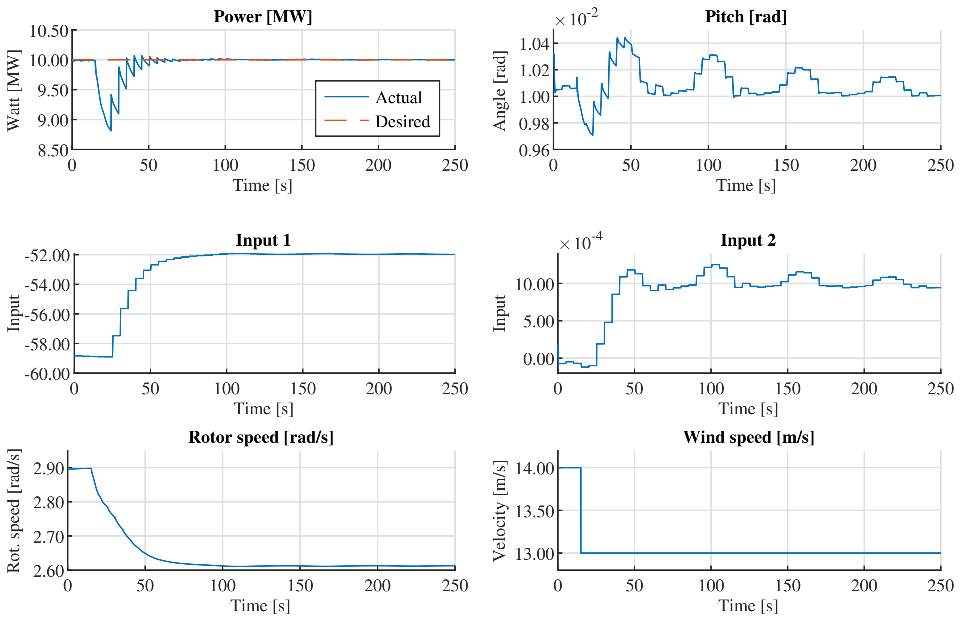

Figure 4.1: System response of linear MPC using full system for predictions, with step inputs to the system.

promising, and gave hope of good performance if the MPC were to control the set points of a PI controller. It did however suffer from long computation times, meaning it could not run in real time. The next step was therefore to reduce the system to fewer states, the results of which are covered in the next section.

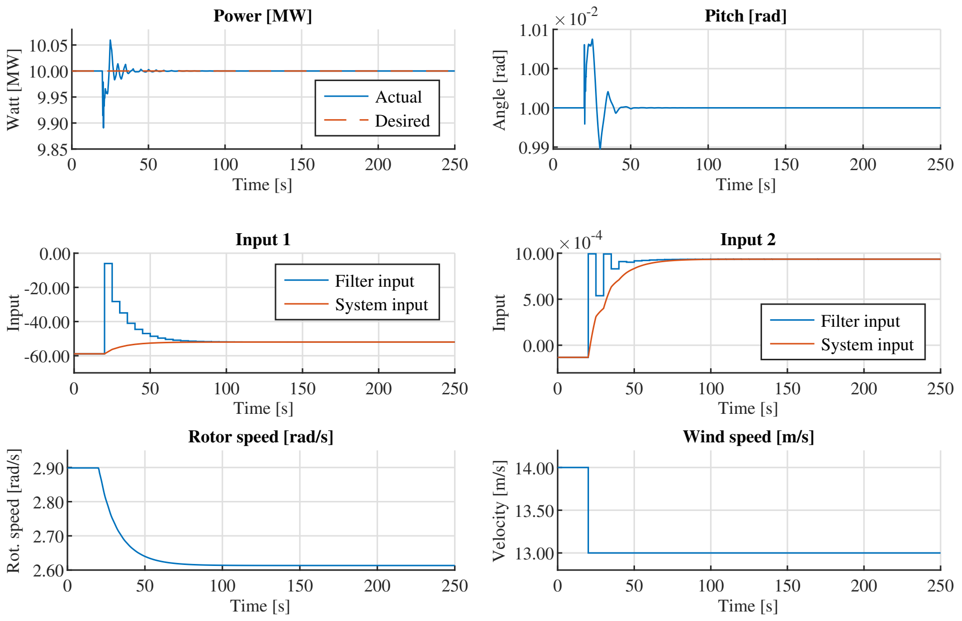

Figure 4.2: System response of linear MPC using full system for predictions, with the MPC controlling filtered inputs to the system.

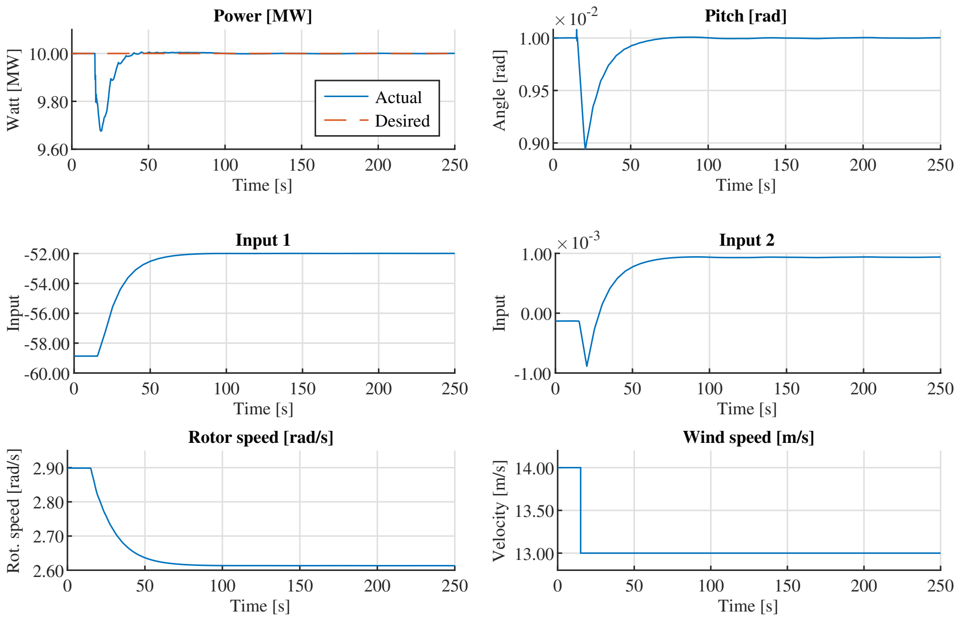

Figure 4.3: System response of linear MPC using full system for predictions, with linearly varying inputs.

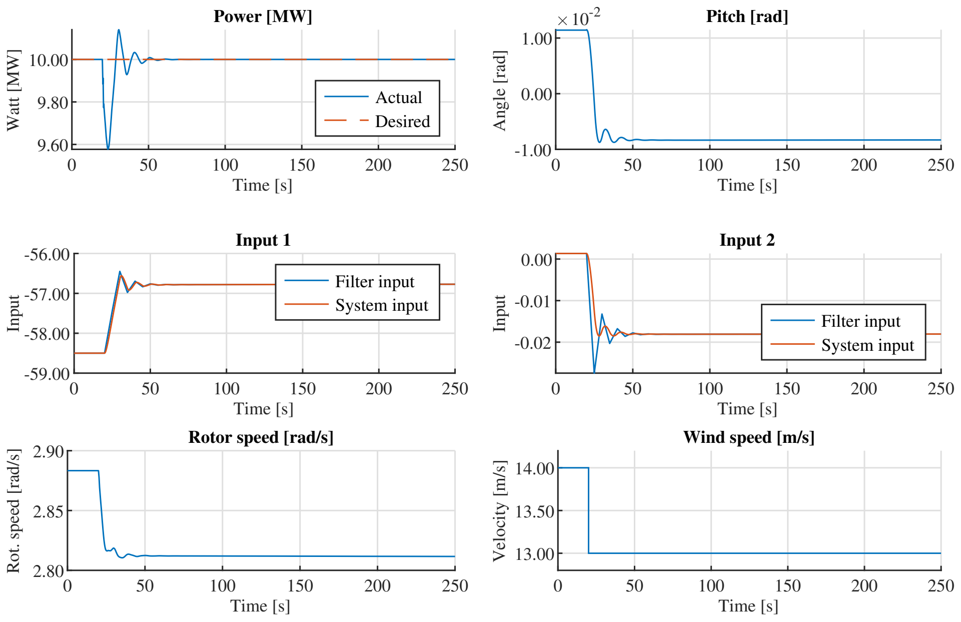

Figure 4.4: System response of linear MPC using full system for predictions, with the MPC controlling linearly changing inputs to a filter. The output of the filter is the input to the system.

4.2 System reduction

To make the system run in real time the linearized system had to be reduced to fewer states. The system was first reduced with balanced reduction, but it could not be reduced to any less than 62 states without loss of power tracking. The system performance with this reduction scheme can be seen in figure 4.5. Note that here the MPC controls a lowpassed input, and two of the states are therefore the internal states of this filter.

The system was then reduced with PCA, resulting in a system that could be reduced down to only 22 states, where two of the states where states of the filter used to lowpass the input. The result can be seen in 4.6. The results can be seen to track the power reference well before the jump in wind speed, but gives a bias after.

To remove the bias stemming from the error in the reduction, the MPC was made to control the set points of a PI controller. This resulted in convergence, as can be seen in figure 4.7.

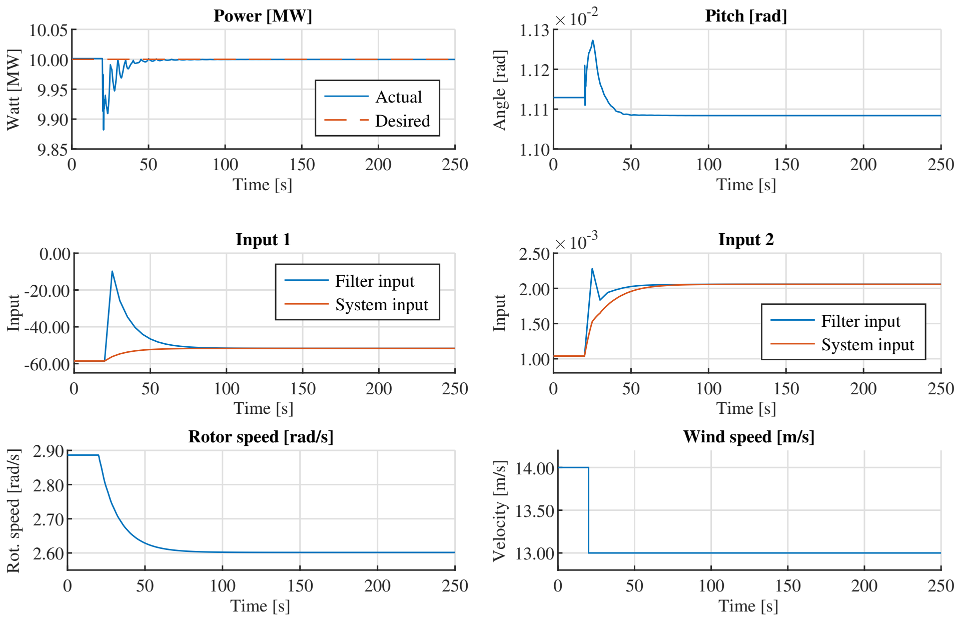

Figure 4.5: System response of MPC with predictions using system reduced to 62 states with balanced reduction. The time horizon is 25 seconds divided into 5 control intervals.

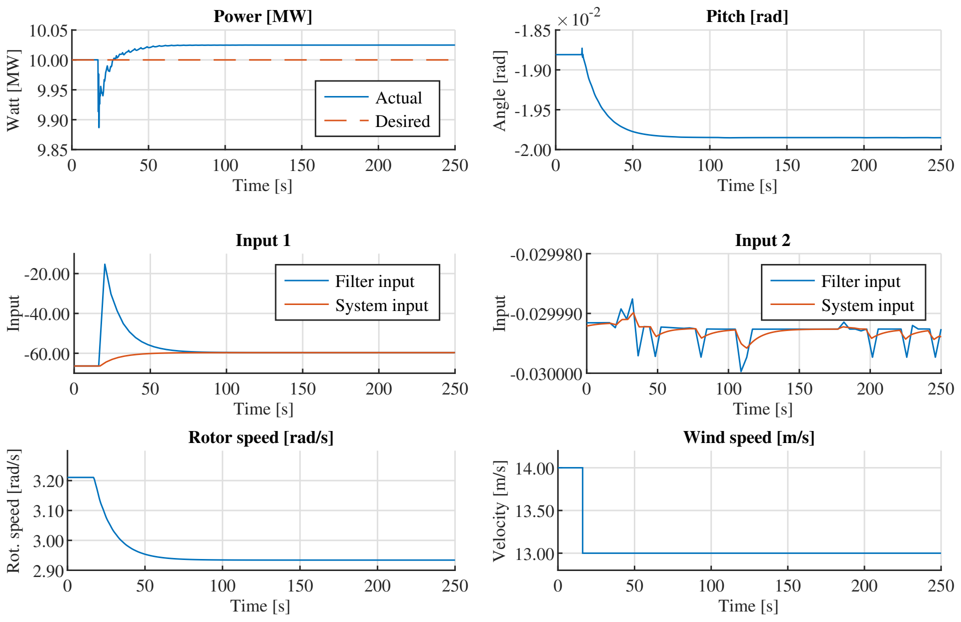

Figure 4.6: System response of MPC with predictions using system reduced to 22 states with PCA. The time horizon is 20 seconds divided into 5 control intervals.

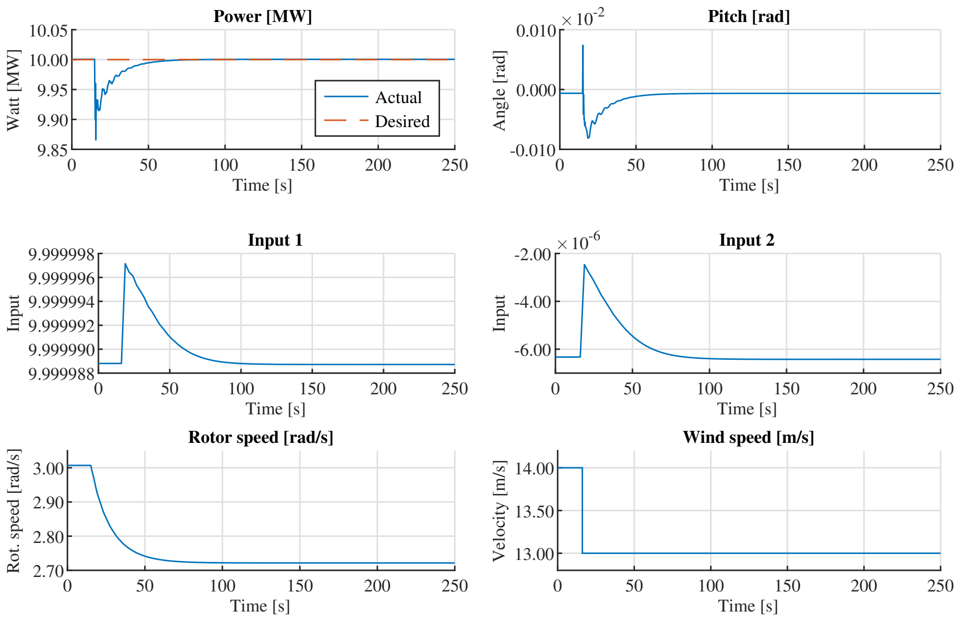

Figure 4.7: System response of MPC with predictions using system reduced to 24 states with PCA, controlling set points of PI controller. The time horizon is 25 seconds divided into 5 control intervals

4.3 Feed forward wind measurements

To predict future wind inputs to the system and adjust the controls ahead of time, LIDAR wind measurements where simulated and used in the MPC's predictions. To better illustrate the behavior of the system and its advantage over a pure PI controller, the wind input is now a ramp function instead of a step. The resulting behavior of a simple PI controller can be seen in figure 4.8, while the response of the MPC controller without feed forward can be seen in figure 4.9. Finally the response of the feed forward MPC can be seen in figure 4.10. As the plots show, the feed forward from the LIDAR measurements lead to a large improvement in power tracking capabilities. The root mean square error between actual and desired power for the three different controllers can be seen in table 4.1, and further illustrate this point.

Controller description	ERMS
Simple PI controller	1.7502
MPC without feed forward	1.7504
MPC with perfect feed forward	0.0979
Feed forward MPC with uncorrelated Gaussian noise	0.3279

Table 4.1: Root mean square error between actual and desired power for different controllers and noise models, showcasing the benefits of LIDAR feed forward.

This performance is assuming the LIDAR measurements of the wind speed are perfect measurements of the mean wind speed of the planes of wind at the measurement distances.

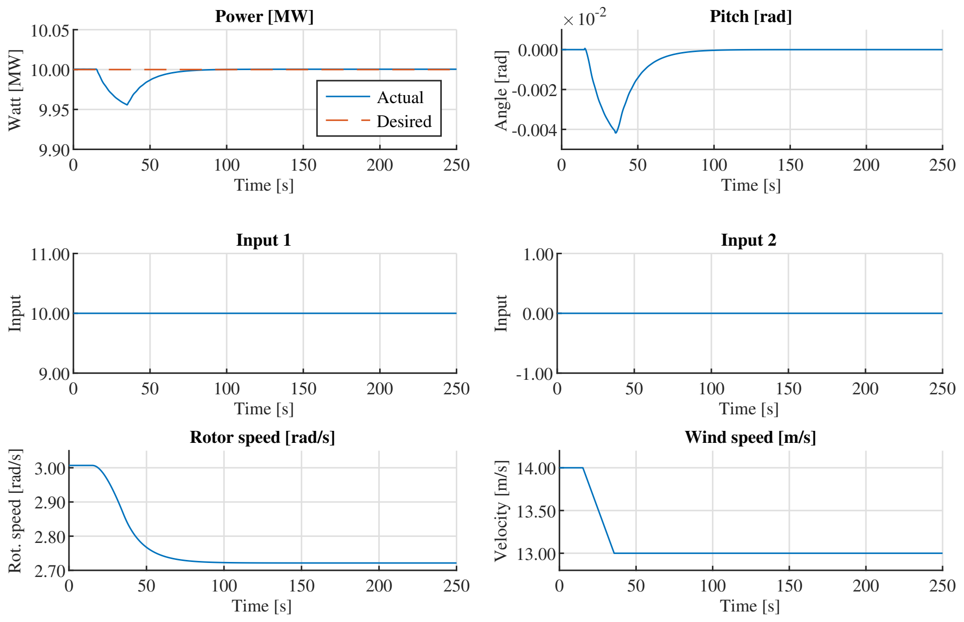

Figure 4.8: PI controller responding to ramp wind input.

In the next step, additive white noise was added to each measurement at each iteration, and the resulting performance can be seen in figure 4.11. The white noise had a power changing linearly between 0.01 at 10 m to 0.02 at 400 m. To counteract the noise the measurements are filtered in the prediction with a lowpass filter with cutoff frequency of 0.1.

Since uncorrelated white noise is not a realistic noise model for turbulent wind speed measurement, a more sophisticated noise model was incorporated, using Finite Difference Method (FDM) techniques to solve the transport equation. A simple PI controller responding to the wind seen in figure 3.2 is plotted in figure 4.12. The same wind input to the system was controlled by a feed forward MPC with simulated LIDAR measurements from a separate FDM velocity state space with Gaussian inlet noise with a power of 1. The velocity state space can be seen in figure 4.13 while the response of the controller is shown in figure 4.14.

Similarily, figure 4.15 contains a plot of a velocity state space with inlet noise with power of 2, process noise with a power of 0.5 and a gust giving a 10 $\mathrm{m/s^2}$ increase to the derivative of the velocity state located 350 m upstream of the rotor, lasting for 4 seconds. The response of the controller when the LIDAR measurements come from this velocity state space is illustrated in figure 4.16.

The RMS error of the pure PI controller and the two variations of the noisy finite difference based measurements can be seen in table 4.2.

The feed forward controller has lower RMS error than the pure PI controller, even in the

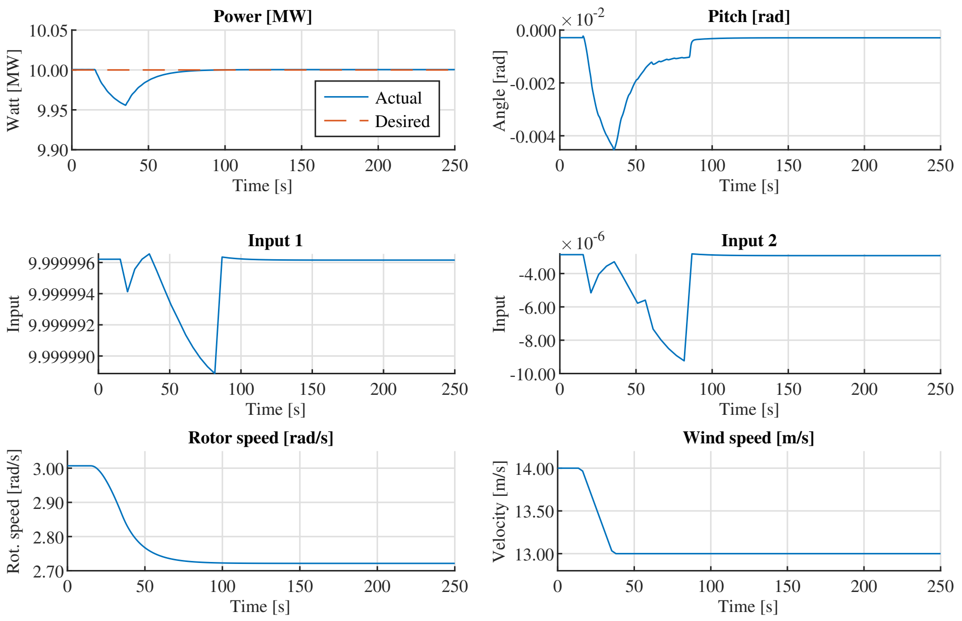

Figure 4.9: MPC without feed forward responding to ramp wind input. The MPC controls PI set points, and has a time horizon of 40 seconds divided into 15 control intervals.

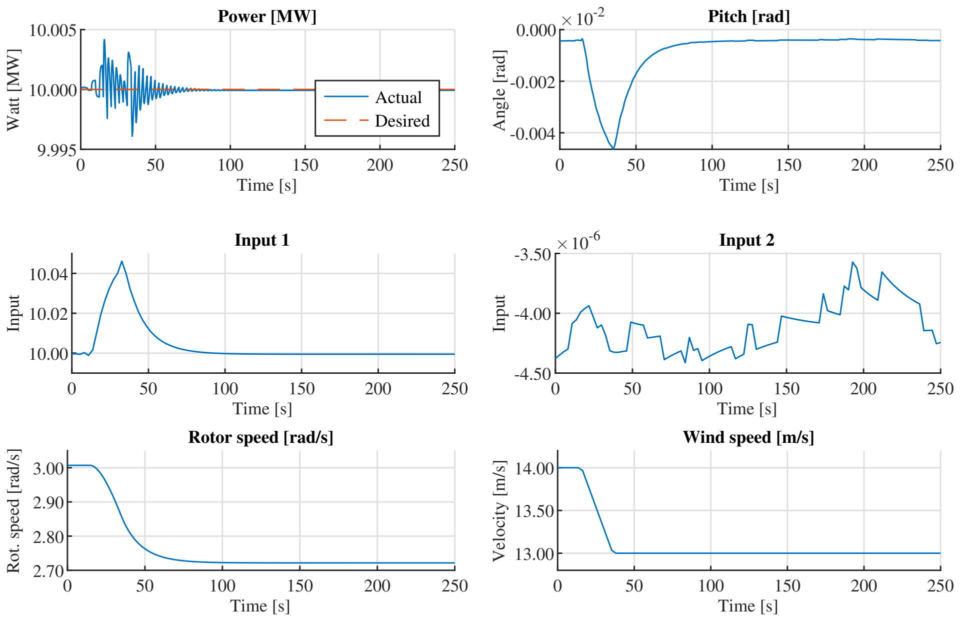

Figure 4.10: Feed forward MPC with noiseless LIDAR measurements responding to ramp wind input. The MPC controls PI set points, and has a time horizon of 40 seconds divided into 15 control intervals.

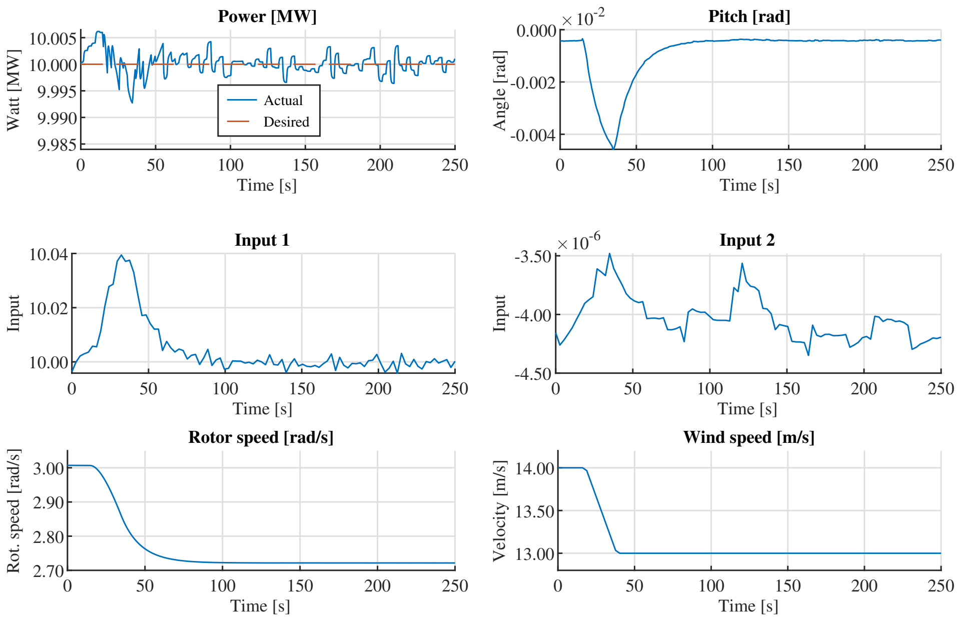

Figure 4.11: Feed forward MPC with LIDAR measurements with additive Gaussian noise, responding to ramp wind input. The MPC controls PI set points, and has a time horizon of 40 seconds divided into 15 control intervals.

presence of gusts, inlett- and process noise on the measured wind state space. Note that the actual input to the system does not have the have these effects.

Controller description	ERMS
Simple PI controller	2.0135
Feed forward MPC with inlet noise	0.3605
Feed forward MPC with gust, inlet and process noise	0.4660

Table 4.2: Root mean square error between actual and desired power, comparing pure PI control over feed forward MPC with finite difference noise.

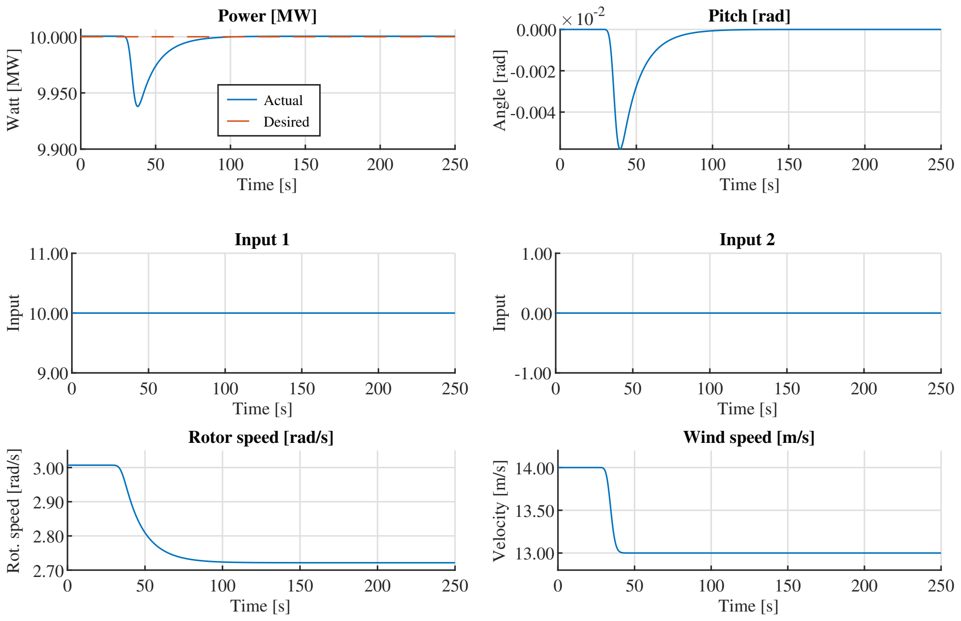

Figure 4.12: Pure PI controller responding to the same input wind as the feed forward MPC controllers seen in figures 4.14 and 4.16.

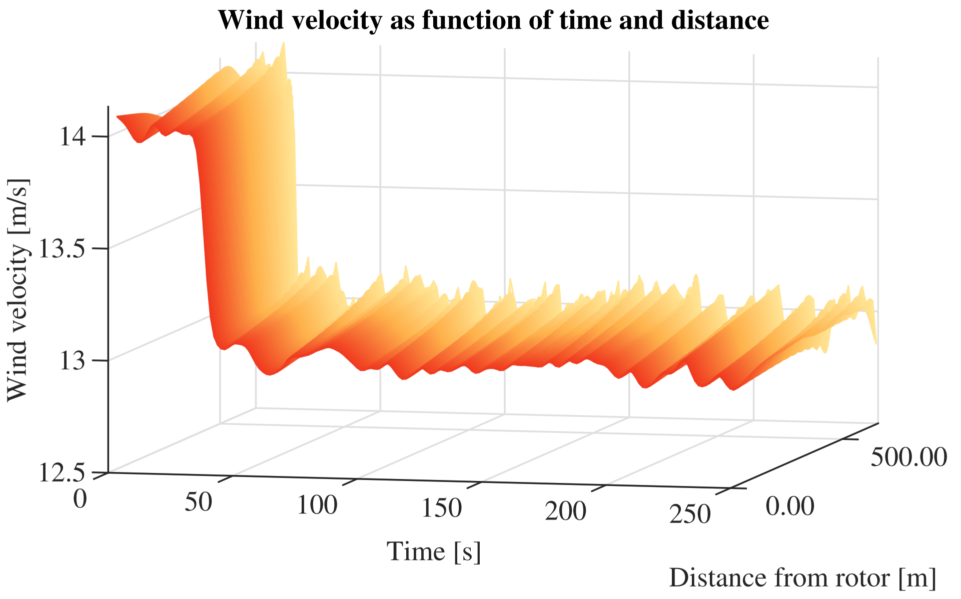

Figure 4.13: Measured wind velocity as a function of distance from rotor and time, with Gaussian noise on inlet. Note that the added Gaussian noise at the inlet of the state space gives correlated noise propagating towards the rotor. This profile is then measured at the 10 measurement distances spaced equally between 10 and 400 meters from the rotor.

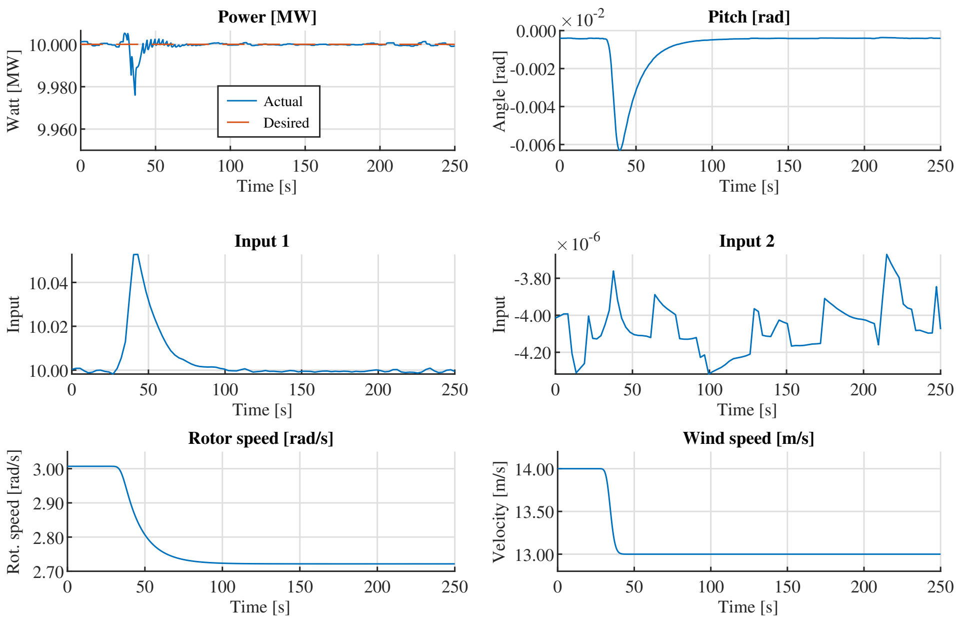

Figure 4.14: Feed forward MPC with LIDAR measurements from the wind profile seen in 4.13.

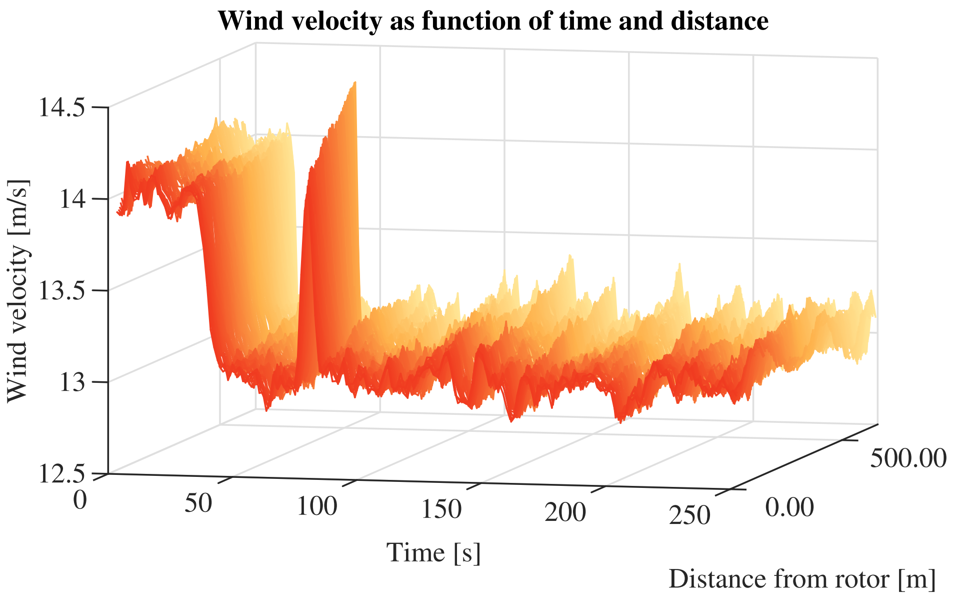

Figure 4.15: Measured wind velocity as a function of distance from rotor and time, with Gaussian noise on inlet, process noise between wind states and an added gust. This profile is measured at the 10 measurement distances spaced equally between 10 and 400 meters from the rotor.

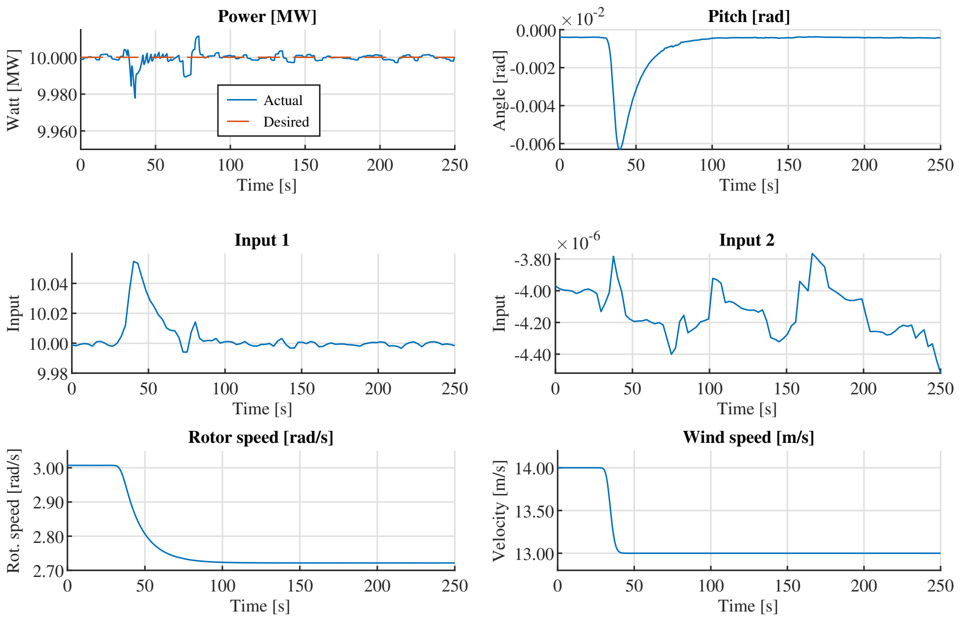

Figure 4.16: Feed forward MPC with LIDAR measurements from the wind profile seen in 4.15.

4.4 Interpolation

In order to better approximate the behavior of the nonlinear system, a parameter-varying system created by interpolation of linearized systems for three different wind speeds was created.

Both of the systems exposed below use feed forward with perfect LIDAR measurements and have a cost function that penalizes a difference between subsequent inputs. They both use a time horizon of 25 seconds divided into 5 control intervals.

4.4.1 Interpolation of full system

The system was linearized for three different wind speeds, and then each element of the system matrices was interpolated separately. The optimal projection matrix was found and applied at each step to the current linearized system, to give a reduced system. This reduced system was then used by the MPC to control the set points of the PI regulator, like before. This resulted in a loss of power reference tracking, most likely because the reduction was not well suited for all operating points, thus making some of the important dynamics be hidden from the MPC's predictions. The result can be seen in figure 4.17. Note that the controller in this case pushed the pitch to its lower limit, which also would have caused the controller to have less operating room if another change in wind would occur.

Using a simple PI controller on the interpolated system gives much better performance, as can be seen in figure 4.18. It did however lead to the rotor speed increasing when the wind went from 14 $\mathrm{m\,s^{-1}}$ to 13 $\mathrm{m\,s^{-1}}$. In all the other scenarios and for all the other controllers the rotor speed has gone down. The author found this very peculiar, but found no other explanation than that it too is an artifact of the interpolation and its lack of precision.

4.4.2 Interpolation of reduced system

Instead of interpolating the full system and using a single projection matrix to reduce the system, the different linearization points where reduced, transformed to a common basis according to the method in section 2.5.2 and then interpolated. Running the MPC on this reduced system with the interpolated full system as the real simulated system gave the results seen in figure 4.19. The RMS power error improved compared to both the MPC with a single projection matrix and to the simple PI controller, but the controller ended up with a small bias. The author does not know exactly why the bias occurred, but it seems plausible that it stems from the interpolation in some manner. The RMS error for the three cases can be seen in table 4.3. This lead to better power tracking than the simple PI controller run on the full interpolated system.

Controller description	ERMS
Simple PI controller	10.6732
Feed forward MPC with interpolated full system and only one reduction	3.7638e+03
Feed forward MPC with interpolated reduced system	6.5850

Table 4.3: Root mean square error between actual and desired power for three different controllers, controlling the interpolated full system.

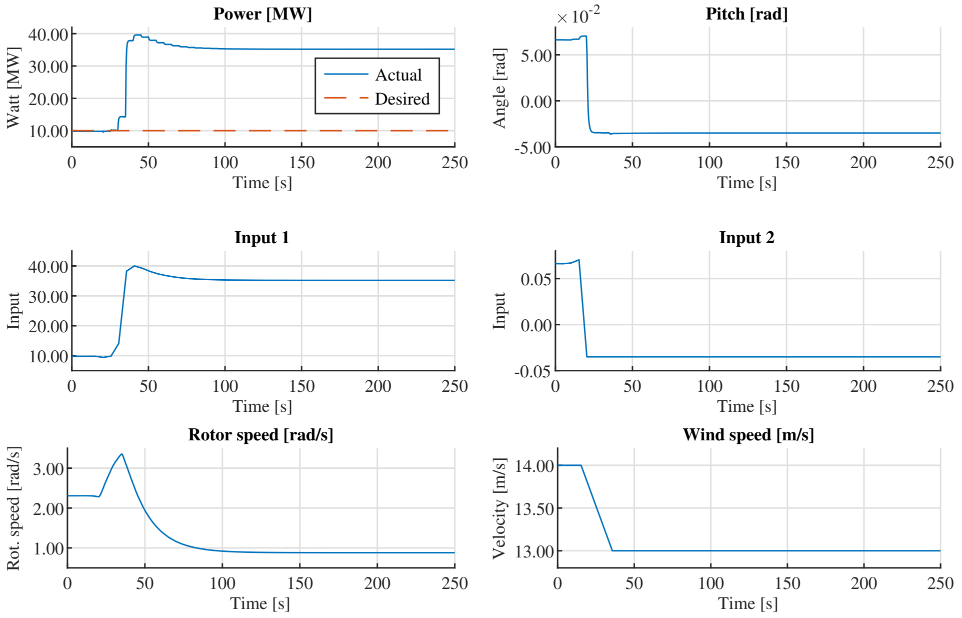

Figure 4.17: System response when running MPC using the interpolated full system with only one static projection matrix for predictions and the interpolated full system as actual system dynamics.

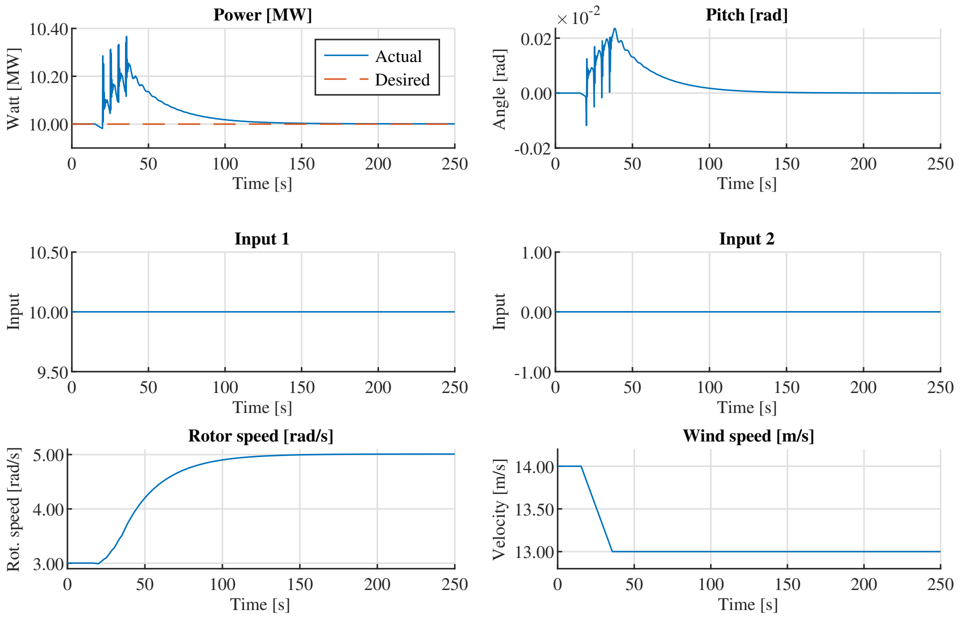

Figure 4.18: System response when using a PI controller on the interpolated full system.

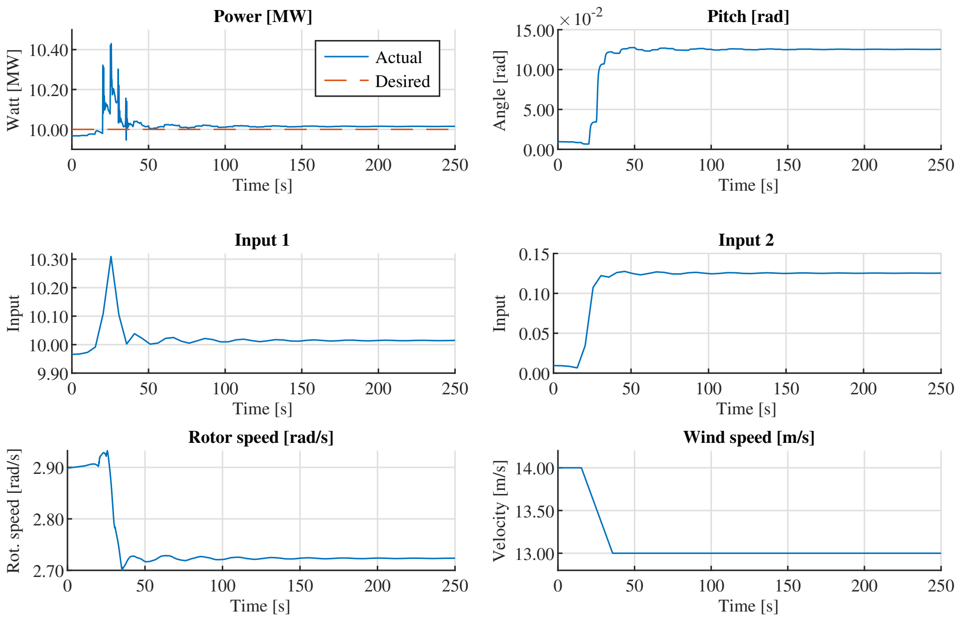

Figure 4.19: System response when running MPC using interpolated reduced system for predictions and interpolated full system as actual system dynamics.

4.5 Moment reduction

To reduce fatigue, it is desirable to reduce the change in the moments of the different parts of the wind turbine. An attempt at making the MPC achieve this was done by putting a cost on the derivative of the moments of the blade roots and the tower nodes. This did however only lead to a slightly faster convergence to zero of the moment derivatives, but made the power tracking much worse. The reason for this might be the fast dynamics of the moments compared to the step size in the MPC, or because of some inherent uncontrollability of the moments in the STAS simulator. The RMS errors between desired and actual states with and without the cost on the moment derivatives can be seen in 4.4.

The moment derivatives and other relevant states for MPC with only a cost on the power error can be seen in figure 4.20, and with a cost on both the power error and the moment derivatives in figure 4.21. Note that a wide range of costs was tried, and the shown performance is with the cost that gave the largest reduction in the RMS value of the moment derivatives. In both these cases the MPC controls lowpass filtered inputs.

State	ERMS with cost	ERMS without cost
Power	35.7951	1.2205
Blade root moment derivative	607.9099	642.4736
Tower moment derivative	1.3149e+04	1.3155e+04

Table 4.4: Root mean square error between actual and desired states with and without a cost on the blade root and tower moment derivatives.

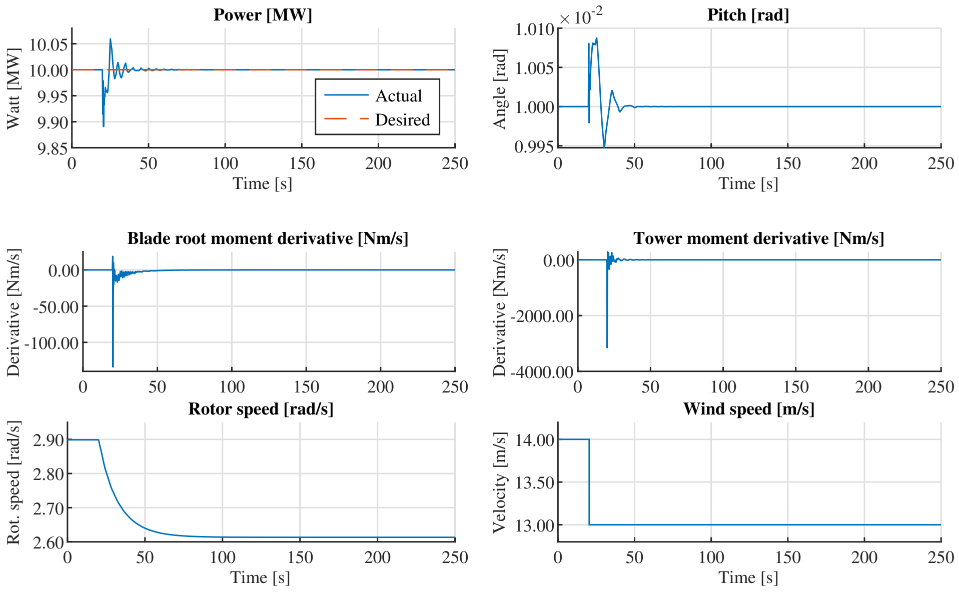

Figure 4.20: System response to a step input in wind without cost on any of the moment derivatives.

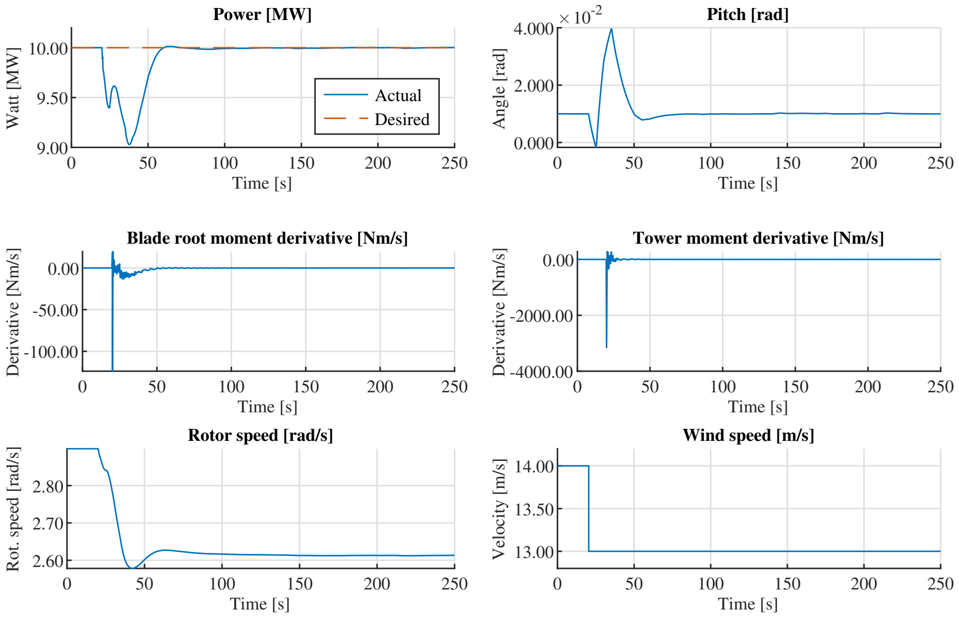

Figure 4.21: System response to a step input in wind with cost on the moment derivatives of the blade root and tower..

Chapter 5

Discussion

The results and the work done for this thesis leads to some interesting observations, which will now be discussed.

5.1 The importance of model reduction

To make the MPC work in real time a reduction of the state space was needed. This did not lead to a noticeable decrease in performance, even when reduced from 103 states to 24. It would seem like a lot of the dynamics are nonessential to the MPC and the control problem at hand. Reducing the system in this manner lead to more computation time available for more advanced control schemes, and was necessary to make real time feed forward MPC with PI set point control possible.

5.2 Controlling PI set points

This leads to the next discussion point, that controlling the PI set points gave better performance than making the MPC control the inputs to the system directly. This is possibly because the PI controller then handles some of the fast dynamics, while the MPC takes care of constraint handling and wind preview using the LIDAR measurements. In most wind turbine applications of MPC this seems like the way to go, as it gets the benefits from both MPC and PI control.

5.3 LIDAR cost/profit analysis

Another subject of discussion is whether or not a LIDAR is worth investing in. It can be rather costly, so it needs to give an economic advantage that outweighs the cost. In the simulation it can be seen that both in the case of perfect measurements, and even in the presence of a non-negligible amount of noise, the feed forward MPC gives substantial decrease in power error. In most real grids a deviation in both the positive and negative direction from the desired power set point leads to a financial penalty. To make a rather simple initial estimate of this penalty, lets assume that it is the same both in the positive and negative case, and set it equal to the price of the lost revenue from selling the power. Assume first that a change in wind input like the one in 4.8 occurs once every minute. This is a rather conservative estimate, as can be understood by looking at for example the real wind speed measurements from the 85 m high wind turbine in figure 13a in [31], where the wind speed clearly fluctuates a lot more than this assumption. The megawatt hours lost from such an event is calculated by

$$\Delta MWh = \sum_{k=1}^{T} abs(P_k - P_{ref})dt, \tag{5.1}$$

where T is the time needed for the error to converge to zero, P_k is the power at time interval k, P_{ref} is the desired power and dt is the time step in hours. Calculating this for the power controlled by the PI controller in figure 4.8 gives a loss of 3.2463e-04 $\mathrm{MW\,h}$. For the noisy feed forward in figure 4.11 this calculation gives 1.1261e-04 $\mathrm{MW\,h}$. Assuming such an event occurs every other minute the loss of energy in a year is 169.7635 $\mathrm{MW\,h}$ for the PI and 59.1878 $\mathrm{MW\,h}$ for the MPC. That is to say, with the noisy feed forward, 169.7635 - 59.1878 = 110.5757 $\mathrm{MW\,h}$ more energy could be sold each year. According to [30] the average energy price in the CWE region (The Netherlands, Belgium, Germany, France and Austria) in 2018 was about 50 euro per $\mathrm{MW\,h}$. Taking this as the electricity price leads to an average increase in revenue

$$\Delta revenue = 110.5757 \times 50 = 5528.8\ euro \tag{5.2}$$

each year. It has proven difficult to get an estimate of the price of nacelle mounted wind turbine LIDARs, since one has to be an actual company owning wind farms for the LIDAR companies to give a price offer. However, assuming it costs around the same as a high performance LIDAR for Simultaneous Localization And Mapping (SLAM) and remote object sensing applications, like the Ouster OS-1 [22], it would cost around 10 000 USD. This means that in two years the LIDAR will have given an increase in profit. And this is assuming such a gust only occurs every minute, and that no other use, like yaw-optimization, can be found for the LIDAR measurements.

5.4 Accuracy of simulated LIDAR measurements

Now that the cost of the LIDAR has been justified assuming the simulation results are correct, a discussion of the accuracy of the simulated LIDAR measurements is needed. First of all, the simulated LIDAR only measures at the distances that the ZX-TM LIDAR [34] measures at, thus it does not have the full future wind input, only a subset of the current wind speed as a function of distance from the rotor. This is, however, better information than a real LIDAR would give, as in a real situation the measurements would also have an error compared to the real wind plane average velocity. This is partially accounted for by the addition of white noise to each measurement, but since the real measurement noise most likely would not have a Gaussian distribution and also be correlated in some manner, the filtering would not work as well as in the simulation. What is worse, the real wind field would change as it approaches the rotor, thus meaning that even with perfect measurement of the instantaneous velocity as a function of distance from rotor the predicted future mean wind speed hitting the rotor would not be exact. An attempt at simulating this effect is what is done with the finite difference noise, but although the MPC proved to handle this kind of noise and thus gives hope that it would work on a real turbine, empirical studies are still needed to verify.

5.5 Discussion of control of LPV system

In the work done for this thesis it was shown that the MPC using an interpolated reduced system gave better performance than the PI controller, when applied to the full interpolated LPV system. This gives hopes that the MPC would work on the full nonlinear STAS model, as the LPV system is a closer approximation of the nonlinear model than the single LTI system. However, it would be a huge benefit to be able to test this on the STAS model directly, so making the STAS model possible to simulate in the time domain should be a main focus if this work is taken further. Also a better interpolation using a larger set of linearized systems would be beneficial.

5.6 STAS not directly suitable for fatigue reduction using MPC

The results of trying to reduce the moment derivatives of the STAS model, as seen in figures 4.21 and 4.20, do not give much hope for fatigue reduction with MPC on the current iteration of the STAS model. The moments are not controllable, and to be able to have an impact on the magnitude of the moment derivatives the goal of power tracking has to more or less be ignored. Even then, the reduction of the magnitude does not seem to reduce the amount of zero crossings of the moment derivatives, which according to [29] is the main cause of fatigue on wind turbines. This leads to the conclusion that in the current state, STAS is not well suited for fatigue reduction using MPC. This might be because of a lack of controllability of the moments in the STAS model or because of the much faster dynamics of the moments compared to the MPCs time discretization. Another plausible reason for the lack of moment control is that STAS might not add the proper moments to the system, maybe since the wind input has been simplified to a single mean wind velocity.

Chapter 6

Conclusion

In this thesis a MPC was developed for STAS, SINTEF's offshore wind turbine simulator. The first iterations used a single linearization of the STAS model, controlling the generator speed and collective pitch of the blades. To achieve better performance, several different types of inputs were researched, from step-inputs, to lowpassed inputs and linearly varying inputs before finally settling on making the MPC control set points of a PI controller, regulating power and pitch.

The main objective for the MPC is to track a desired power output, but an attempt at reducing the fatigue through minimizing the moment derivatives was also done. This did however not prove promising, possibly because of a lack of controllability of the moments in the STAS model or because the moments have faster dynamics than the time-discretization of the MPC.

To speed up computation time and give room for more advanced control schemes, the model was reduced. First it was reduced from 103 to 62 states with balanced reduction and then down to 24 states with PCA. This made it possible to make more predictions each step while keeping the system real time capable.

A large improvement in power tracking error was seen when feed forward MPC was implemented using simulated LIDAR measurements, following the specifications of a ZX-TM nacelle mounted LIDAR. It lead to a fivefold reduction in RMS error even in the presence of a non-negligible amount of additive gaussian noise on the measurements. Another noise model was also implemented, using a finite difference scheme to simulate correlated noise that changes as it approaches the rotor. Also in this case the system gave a substantial reduction in the RMS error. A subsequent analysis of the cost and benefits of investing in a LIDAR showed that even in a rather conservative estimate the LIDAR should pay for itself within two years.

Finally a LPV system was created from interpolations of several linearizations of the STAS model. The system did not behave well when only one static projection matrix was used to reduce the system in all operating conditions. Better performance than the PI controller was achieved when each linearization was reduced separately and transformed into a form where they could be interpolated. The resulting reduced size LPV system proved to be good enough for predictions to give a lower RMS power error than the PI controller, when using the full interpolated system as the simulated real system.

The above results and discussion leads the author to conclude that MPC, especially when coupled with LIDAR wind measurements, is a promising subject of research for better power tracking and energy capture in offshore wind turbines, as well as to achieve grid stability in a world with higher and higher penetration of renewable energy.

Chapter 7

Future work

This section covers what could possibly be done in the future if the work of this thesis is to be continued.

Turbines have several operating regions [3], with different goals and constraints in the different regions. The work in this thesis focuses on the operating region where the turbine is at full load, but below rated rotor speed. A logical extension of the current system would be to have it operate at more than just this region. This would require different cost functions and constraints in the different regions, and smart switching between these. This is not an insurmountable task, as it has been done before [12] for other models, and is a crucial step towards real turbine application.

Another important next step is to create a state estimator. Not all states are directly measurable, and those that are have some kind of measurement error, so to use the full state in the MPC prediction, a state estimator is needed.

Since the MPC developed for this thesis did not prove very promising at reducing the fatigue of the turbine, it seems promising to couple the MPC with a low-level PI controller to dampen drive train and individual pitch control to reduce fatigue from tower shadow and wind speed height gradient. This way one would get the benefits of constraint handling and wind preview of the MPC, while the fast fatigue-causing dynamics can be handled by the PI.

Yet another future improvement would be to make the STAS model possible to simulate in the time domain, not just the frequency domain. This would give a much better validation of the system and expose potential errors stemming from e.g. the way the LPV system has been created. The step after this would then be to test the system on a real wind turbine, to weed out modelling errors and validate if the system would work outside of a simulated environment. An alternative to simulating with STAS in the time domain would be to validate the controller on a simulator already capable of doing time-simulation, like Ashes [2] or FAST [28].

Another improvement to STAS, that actually is currently on the way, is to couple it with a flow model. This would give more realistic inputs, moments and power response and would be one step closer to the real world.

Finally another vein of research the author would want to investigate is to make a first-principle model of the turbine with as few states as possible and check if one could get good performance controlling the STAS model using a MPC predicting with linearizations of this new model.

Bibliography

[1] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, In Press, 2018.

[2] Simis as. Ashes. `https://www.simis.io/`, 2019. [Online; accessed November 26, 2019].

[3] Fernando Bianchi, Hernán De Battista, and Ricardo Julian Mantz. *Wind Turbine Control Systems: Principles, Modelling and Gain Scheduling Design*. Springer, 01 2007.

[4] Antoine Borraccino, David Schlipf, Florian Haizmann, and Rozenn Wagner. Wind field reconstruction from nacelle-mounted lidars short range measurements. *Wind Energy Science Discussions*, pages 1–, 03 2017. doi: 10.5194/wes-2017-10.

[5] Jan Caigny, Rik Pintelon, J. Camino, and Jan Swevers. Interpolated modeling of lpv systems. *IEEE Transactions on Control Systems Technology*, 22:2232–2246, 11 2014. doi: 10.1109/TCST.2014.2300510.

[6] Alessandro Castagnotto, Maria Cruz Varona, Lisa Jeschek, and Boris Lohmann. Sss & sssmor: Analysis and reduction of large-scale dynamic systems in matlab. *at - Automatisierungstechnik*, 65, 01 2017. doi: 10.1515/auto-2016-0137.

[7] I. C. Cheeseman. Helicopter theory. w. johnson. princeton university press. 1980. 1090 pp. illustrated. £52.90. *The Aeronautical Journal (1968)*, 85(842):123–123, 1981. doi: 10.1017/S0001924000029523.

[8] Moritz Diehl. Numerical optimal control, 2011.

[9] Equinor. Equinor—the world's leading floating offshore wind developer. `https://www.equinor.com/no/what-we-do/hywind-where-the-wind-takes-us.html`, 2019. [Online; accessed Desember 11, 2019].

[10] BVG Associates for NORWEP. Annual global offshore wind market report 2019, 2019.

[11] KEITH GLOVER. All optimal hankel-norm approximations of linear multivariable systems and their l^∞-error bounds†. *International Journal of Control*, 39(6): 1115–1193, 1984. doi: 10.1080/00207178408933239. URL `https://doi.org/10.1080/00207178408933239`.

[12] L Henriksen. Model predictive control of a wind turbine with constraints. *European Wind Energy Conference and Exhibition 2008*, 5, 01 2008.

[13] Andrzej Holdyk Karl Merz, Konstanze Kölle. An electromechanical model of the totalcontrol reference wind power plant, 2019.

[14] Virginia Klema and Alan Laub. The singular value decomposition: Its computation and some applications. *IEEE Transactions on automatic control*, 25(2):164–176, 1980.

[15] Arne Koerber and Rudibert King. Combined feedback–feedforward control of wind turbines using state-constrained model predictive control. *IEEE Transactions on Control Systems Technology*, 21:1117–1128, 2013.

[16] Jason Laks, Lucy Pao, Eric Simley, Alan Wright, Neil Kelley, and Bonnie Jonkman. Model predictive control using preview measurements from lidar. 01 2011. ISBN 978-1-60086-950-1. doi: 10.2514/6.2011-813.

[17] D. Medici, S. Ivanell, J.-Å. Dahlberg, and P. H. Alfredsson. The upstream flow of a wind turbine: blockage effect. *Wind Energy*, 14(5):691–697, 7 2011. ISSN 1099-1824. doi: 10.1002/we.451. URL `https://doi.org/10.1002/we.451`.

[18] B. Moore. Principal component analysis in linear systems: Controllability, observability, and model reduction. *IEEE Transactions on Automatic Control*, 26(1):17–32, February 1981. ISSN 2334-3303. doi: 10.1109/TAC.1981.1102568.

[19] S. Oertel, M. Eggert, C. Gutsmuths, P. Wilhelm, H. Müller, and H. Többen. Validation of three-component wind lidar sensor for traceable highly resolved wind vector measurements. *Journal of Sensors and Sensor Systems*, 8(1):9–17, 2019. doi: 10.5194/jsss-8-9-2019. URL `https://www.j-sens-sens-syst.net/8/9/2019/`.

[20] K. Ostergaard, Per Brath, and J. Stoustrup. Estimation of effective wind speed. *Journal of Physics, Conference series: The Science of Making Torque from Wind*, 75, 06 2007. doi: 10.1088/1742-6596/75/1/012082.

[21] Heiko Peuscher, Jan Mohring, Rudy Eid, and Boris Lohmann. Parametric model order reduction by matrix interpolation. *Automatisierungstechnik*, 58:475–484, 08 2010. doi: 10.1524/auto.2010.0863.

[22] The Robot Report. Ouster releases os1-32 high-resolution, 32-channel lidar sensor. https://www.therobotreport.com/ouster-releases-osi-32-high-resolution-32-channel-lidar/, 2019. [Online; accessed November 25, 2019].

[23] David Schlipf, Dominik Johannes Schlipf, and Martin Kühn. Nonlinear model predictive control of wind turbines using lidar. *Wind Energy*, 16(7):1107–1129, 2013. doi: 10.1002/we.1533. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/we.1533.

[24] Florian Selot, Daniel Fraile, Guy Brindley, and Colin Walsh. Offshore wind in europe: Key trends and statistics 2018, 02 2019.

[25] Eric Simley, Lucy Pao, Rod Frehlich, Bonnie Jonkman, and Neil Kelley. Analysis of wind speed measurements using continuous wave lidar for wind turbine control. *49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, 01 2011. doi: 10.2514/6.2011-263.

[26] SINTEF. Eera deepwind'2020, trondheim, norway 15 - 17 january 2020. https://www.sintef.no/projectweb/eera-deepwind/, 2019. [Online; accessed December 11, 2019].

[27] Martin D. Spencer, Karl A. Stol, Charles P. Unsworth, John E. Cater, and Stuart E. Norris. Model predictive control of a wind turbine using short-term wind field predictions. *Wind Energy*, 16(3):417–434, 2013. doi: 10.1002/we.1501. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/we.1501.

[28] Michael A. Sprague, Jason M. Jonkman, and Bonnie Jonkman. *FAST Modular Framework for Wind Turbine Simulation: New Algorithms and Numerical Examples*. 2015. doi: 10.2514/6.2015-1461. URL https://arc.aiaa.org/doi/abs/10.2514/6.2015-1461.

[29] Herbert Sutherland. On the fatigue analysis of wind turbines. 6 1999. doi: 10.2172/9460.

[30] Tennet. Annual market update 2018. https://www.tennet.eu/news/detail/electricity-prices-on-european-wholesale-markets-increased-in-2018/, 2019. [Online; accessed November 25, 2019].

[31] Takanori Uchida. Computational investigation of the causes of wind turbine blade damage at japan's wind farm in complex terrain. *Journal of Flow Control, Measurement and Visualization*, 06:152–167, 01 2018. doi: 10.4236/jfcmv.2018.63013.

[32] Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, Mar 2006. ISSN 1436-4646. doi: 10.1007/s10107-004-0559-y. URL https://doi.org/10.1007/s10107-004-0559-y.

[33] Thomas Wolf, Heiko Peuscher, and Boris Lohmann. Model order reduction by approximate balanced truncation: A unifying framework. *at-Automatisierungstechnik*, 61:545–556, 08 2013. doi: 10.1524/auto.2013.1007.

[34] ZXLidars. Zx tm – turbine mounted wind lidar. `https://www.zxlidars.com/wind-lidars/zx-tm/`, 2019. [Online; accessed October 3, 2019].

Glossary

Ashes A design and analysis software for onshore and offshore wind turbines. 54

CasADi An open-source tool for nonlinear optimization and algorithmic differentiation. 11

CPU Central Processing Unit. 25

FAST National Renewable Energy Laboratory's primary computer aided engineering tool for simulating the coupled dynamic response of wind turbines. 2, 54

IPOPT A software library for large scale nonlinear optimization of continuous systems, stands for Interior Point OPTimizer. 23

MATLAB A multi-paradigm numerical computing environment and proprietary programming language developed by MathWorks. 15

nlpsolver CasADi's nonlinear program solver function. 23

RAM Random Access Memory. 13

SINTEF Selskapet for industriell og teknisk forskning ved Norges tekniske høgskole. 1–3, 13, 32

sssMOR MATLAB library for sparse state space model reduction. 25

STAS The wind turbine simulator developed at SINTEF. 1–4, 13, 14, 23, 24, 26, 31, 47, 51, 52, 54, 55

tbr Function in sssMOR MATLAB library for creating a balanced realization from a sparse system. 25

ZX LIDARS Company making LIDARs for wind turbines. 26

ZX-TM Nacelle LIDAR from ZX LIDARS. 26, 28, 50, 52

List of Acronyms

BEM Blade Element Momentum. 14

FDM Finite Difference Method. 39

LIDAR LIght Detection And Ranging. 2, 3, 9, 23, 25, 26, 28–31, 38, 39, 44, 49, 50, 52, 53

LPV Linear Parameter-Varying. 2–4, 17, 19, 22, 23, 25, 31–33, 51–54

LTI Linear Time-Invariant. 4, 17–19, 22, 23, 31, 32, 51

MBS Multibody System. 7

MPC Model Predictive Control. 1–4, 9, 14, 23–26, 28, 29, 32–34, 36, 38, 39, 44, 45, 47, 49–55

NLP NonLinear Program. 11–13, 23, 24

PCA Principal Component Analysis. 2, 3, 15, 16, 25, 32, 36, 52

PI Proportional-Integral. 2, 3, 24, 25, 29, 33, 34, 36, 38, 39, 44, 45, 49–54

RADAR RAdio Detection And Ranging. 25

RMS Root Mean Square. 2, 3, 25, 39, 45, 47, 52, 53

SLAM Simultaneous Localization And Mapping. 50

SVD Singular Value Decomposition. 15, 16, 21, 25

TSR Tip Speed Ratio. 6